August 2004

SERVO MAGAZINE

Along Came PolyBot

Jurassic PARC

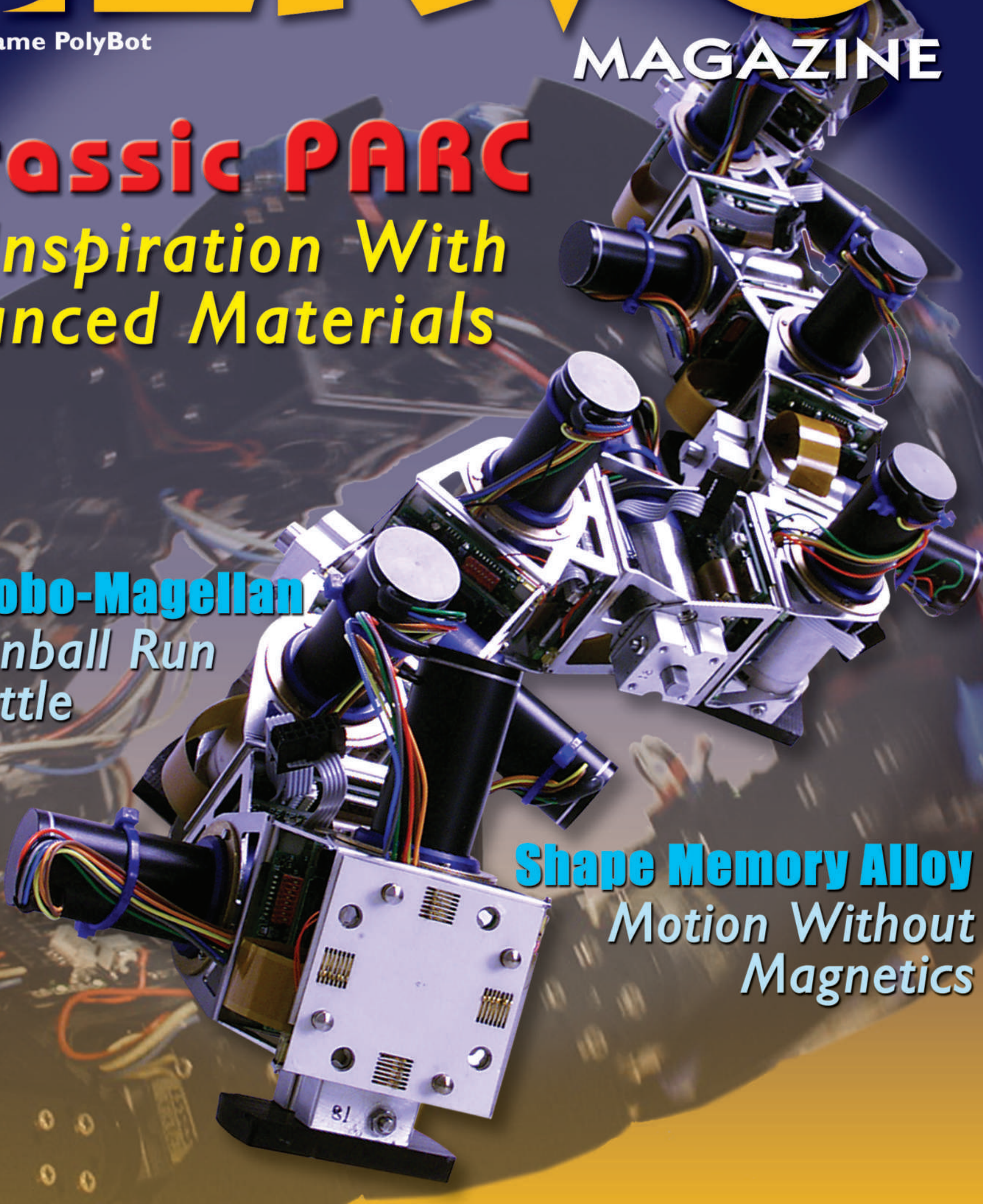
Bio-Inspiration With Advanced Materials

SRS' Robo-Magellan

Cannonball Run In Seattle

Shape Memory Alloy

Motion Without Magnetics



TETSUJIN 2004
Man joins Machine in a test of Imagination, Physical Strength, and Mental Agility.
October 21-23
At RoboNexus
Santa Clara, CA
100 LBS
www.servomagazine.com
Enter the competition!

08.2004

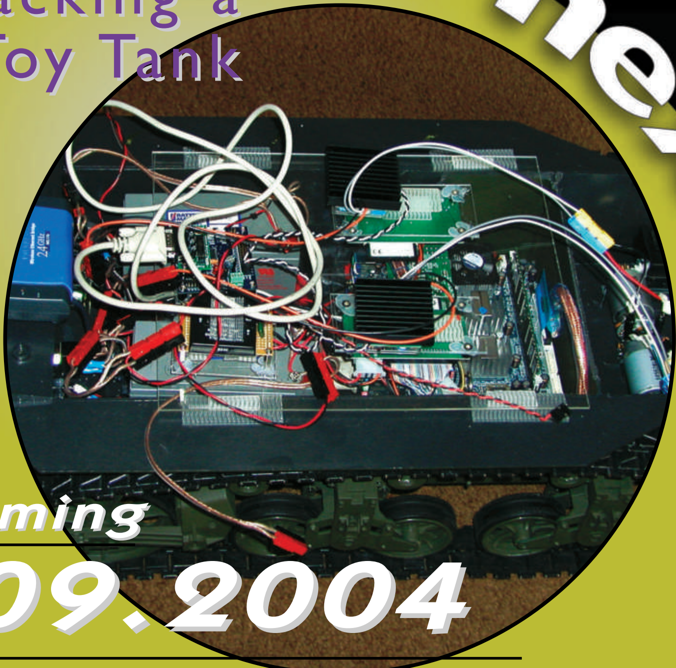
Vol. 2 No. 8

SERVO

table of contents

In our next issue

Hacking a Toy Tank



Coming 09.2004

SERVO Magazine (ISSN 1546-0592/CDN Pub Agree#40702530) is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. APPLICATION TO MAIL AT PERIODICALS POSTAGE RATE IS PENDING AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, 430 Princeland Court, Corona, CA 92879-1300** or Station A, P.O. Box 54, Windsor ON N9A 6J5.
cpcreturns@servomagazine.com

columns

8 Mind/Iron
14 Rubberbands
24 Ask Mr. Roboto
38 Menagerie
39 Robotics Resources
46 GeerHead
62 Robytes
72 The Assembly Line
75 Tetsujin Tech
78 Lessons from the Lab
81 Appetizer

departments

4 Tetsujin Info
8 Publisher's Info
9 Bio-Feedback
34 New Products
64 Events Calendar
65 Robotics Showcase
76 *SERVO* Bookstore
82 Advertiser's Index

features

10 **BEAM Me Up, Photovore!**
by Thomas Gray and J. Wolfgang Goerlich

18 Armed, Wheeled, and Powered
by Tom Carroll

29 Circumnavigating the Seattle Center
by Michael Miller

43 Engineered to Perfection
by Allan Comeau, Ph.D.

52 Shaping Up with SMAs
by Don Wilcher

57 An Analog Sensor Core
by D. Jay Newman

68 Polly Want a PolyBot?
by Dan Danknick

Mind / Iron

by Dan Danknick

Recently, I went in for a casting interview for the TV show *Monster House.* I'd done an episode of *Monster Garage* a few years ago and my name popped up as a potential candidate for one of their transmogrification projects.

As I walked through the production studios on my way out, I took notice of the long line of offices dedicated to current — and upcoming — TV shows. Each of them had the same basic theme: build something cool by the end of the show to win the prize.

Television is a cutthroat business and, if a show doesn't pull in the ratings, it gets yanked. The success of this genre is telling and I believe it reveals an important facet of the human mind; in this age of pre-made *everything,* we still like to build things.

I've written in earlier columns stating that competition spurs innovation. Holding a competition — either against the clock or against others — is the spice of the hobbyist experience. This is fun stuff we think about on the drive home from work and spend our Saturday mornings working on in the shop. That is why we actively work to encourage and sponsor competitions here at *SERVO Magazine.*

We currently have our robotic manipulators in three events. The first is Tetsujin 2004, the powered exoskeletal weight lifting competition we're holding at RoboNexus in October of 2004. On page 75, you can check out two of the teams that have stepped up to the plate — er bar — for this groundbreaking event. They are embarking on something that has never been done before and, hopefully, their enthusiasm will pop right off the page.

Second is the SRS/*SERVO Magazine* Robo-Magellan competition we announced last month. The Seattle Robotics Society came up with this event in the wake of the DARPA Grand Challenge and I think it's a superb contest that every serious robot builder should consider. On page 29, you can read about contestant Michael Miller's practical analysis of the course and the robot he's planning to field.

Third, we are announcing a new competition this month on page 71 — Hack-a-Sapien. After playing with a Robosapien for a few minutes, I concluded that this would be a really fun toy to start modifying. I also started receiving questions from readers about publishing hacks in response to Nick Blye's two part article on the RS in the May and June 2004 issues. So, the contest was born to address all of these items at once! I'm even in touch with RS creator Mark Tilden, who has tossed some interesting tidbits out to aid the competitors. Visit the contest command center on our website, **www.servomagazine.com/hack-a-sapien/** and check out the buzz.

According to our online survey, over 80% of you can program a microcontroller. That means you can enter both Robo-Magellan and Hack-a-Sapien! These contests are designed to accommodate your skills and, unashamedly, to offer you an intellectual challenge. More importantly, as Dr. Allan Comeau discusses in his feature on human perfection, one of our goals should be continued learning and improvement. Who knows — you might one day end up as the host of a new TV show, *Monster Robot!* SV

Published Monthly By
The TechTrax Group — A Division Of
T & L Publications, Inc.
430 Princeland Court
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
www.servomagazine.com

Subscription Order ONLY Line
1-800-783-4624

BIO-FEEDBACK

Dear *SERVO*,

I was glad to see John Rigg's Machine Man Band mentioned in "Menagerie." This guy is an amazing collector and builder of robots. John would make a terrific contributor to *SERVO Magazine.* He has several thousand antique and modern robots in his collection and has constructed The Robot Hut museum near his home in Spokane, WA. I'll let his website do the talking: **www.robothut.robotnut.com/** Don't forget to check out his projects section.

Gary Kaminski
via Internet

Dear *SERVO*,

Thanks to you and Gordon McComb for including our **RobotStore.com** site in the June 2004 "Robotics Resources" column (page 76). Readers should note, however, that the description given for the OctoBot as a solar powered, self-learning robot should actually be "a battery powered, self-recharging robot." Solar robots are great, but we developed the OctoBot to give experimenters a robot that can recharge on its own whenever it needs to — day or night! Keep up the great work!

Roger G. Gilbertson
President
Mondo-tronics, Inc.

ERRATA

Roger Gilbertson's article "From Mars to Your Window Sill" (June 2004) contains some errors due to Greek characters becoming lost in translation. On page 65, the circuit parts list, items 1, 2, and 3, should be 150 ohms, 100K ohms and 1-20K ohms. The labels for those same parts in Figure 3 should also be marked as "ohms" rather than "W." Finally, item AA1 should read "Muscle Wire, 150 micrometer diameter, 10 cm long." Kudos! (Greek for "thank you.")

Dear *SERVO*,

I am a U.K. subscriber to your magazine and I enjoy reading the various articles on robotics. I note you did an article on metal working — bending, shaping, etc. — in the *Amateur Robotics Supplement #2* (August 2003). I would like to know if your magazine would write further on this topic and other mechanical areas of robotics, as this side of the hobby particularly interests me. Also, would you consider doing a feature on PLCs, which feature largely in industrial automated machines?

Simon Griffiths
via Internet

Dear *SERVO*,

I'd like to see more OOPic related articles. A lot of robot building friends are using OOpics over the other controllers out there, like Stamps, etc. Perhaps list your "New Products" on your website, as well. Otherwise, I enjoy your magazine.

Darrell Toland
Seattle, WA

Dear *SERVO*,

Even the newest arm from Schilling Robotics reaches for *SERVO Magazine* when it can! Thanks to Jeff Kroll for the cool picture!

After reading the July 2004 "GeerHead" column, I was wondering if David Geer or anyone else would be able to track down a robot that was the winner of a *Robotics Age Magazine* contest. An article about this machine —named AVITAR — was published on page 22 of the Jan/Feb 1982 (Vol. 4, #1) issue of that now defunct publication.

In the article, Charles Balmer, Jr. (AVITAR's creator) made a great comment about the naming of a robot that has always stuck with me: *"A robot is somewhat like a child. It requires a great deal of patience, time, and energy to construct and, then, as it limps and crashes and smokes its way to adulthood, we as mothers and fathers learn something about being a robot, while — hopefully — our robot learns something about being human. (Gak! — Editor Dan) If for no other reason than to have something to yell in a fit of frustration and anger!"*

I am sure I am not the only person who knows just what he meant. I have often wondered about what became of AVITAR and if Mr. Balmer is still building robots. Thank you for the best magazine since *Robotics Age*.

Clifford Boerema, Jr.
via Internet

Announcing Our New Area Code

As of July 17, our area code will change from (909) to (951). This will affect both our phone and fax numbers.

got bot?

Whether you have a build, code, or theory to share, *SERVO* wants to know what you — the resident of the robot workshop — are creating. We want you to Email us your article submissions. Some topics of interest are:

- Sensors and signal processing
- Mechanical fabrication
- Software techniques
- Data protocols
- Unique drive geometries
- Material selection and use
- Distributed communication

BEAM Robotics Step by Step PART 1

An Intro to BEAM and the BBPV

by Thomas Gray and J. Wolfgang Goerlich

BEAM — an acronym for Biology, Electronics, Aesthetics, and Mechanics — is a design philosophy centered on minimizing part count, power usage, cost, and — above all — complexity. At its simplest, BEAM uses recycled electronic and mechanical parts to create amusing little robots that can mimic basic biological behaviors, such as phototropism (responding to light) or thermotropism (responding to heat). In its more complex forms, it takes cues from biology to solve electromechanical problems.

For instance, most organisms walk using bundles of nerves that oscillate to create patterns of movement. By using biomimicry or creating simplified models of complex biological systems, we can design a robot that uses as few parts as possible, yet demonstrates complex behavior patterns.

Being cheap and versatile, BEAM designs are useful for hobby robotics and virtually anything where low cost per robot is more important than precision or programmability. Exactness is not a strength of BEAM (or of most living organisms), so you will not see a BEAM robot doing factory work. Thus far, BEAM designs have found niches in micro-machines, swarms, and toys. Depending upon the development of the technology, BEAM may also find Nanotech applications.

Having a Lot of Nerve

The BEAM Nervous Net mimics the biological equivalent to give the BEAM robot some measure of autonomy. The basic unit, which we will touch on briefly in this article and examine in a bit more depth next month, is the "nervous neuron" or Nv. Now, real neurons are complex and perform several diverse functions. Turning again to biomimicry, we need to simplify things a bit. For our purposes, BEAM neurons simulate real neurons in that they have a threshold before activating and then are active for a specific amount of time.

Most organisms rely on central pattern generators (CPGs) to coordinate their limbs into gaits. We combine BEAM neurons into oscillators or Nervous Nets to mimic CPGs.

Depending upon the purpose, a Nervous Net could be made from transistor circuits, plain and Schmitt inverters, op-amps, toy blob chips, or they could even be simulated on a microprocessor. In keeping with the minimalist spirit, the Nervous Nets we describe here use inverters because such designs require fewer parts, need less power, and are generally simpler.

Step 1: Playing With Inverters

In this article, we present a typical BEAM circuit based on a single octal inverter IC and a simple robot based on it. An inverter (Figure 1) is an electronic device used to invert the logic level input to it — that is, it switches the logic "high" to

Mark Tilden

Mark Tilden is to BEAM Robotics as Linus Torvalds is to Linux. Tilden — who began building robots in the 1980s — coined the term BEAM and patented his electronic Nervous Net in 1994 while at the University of Waterloo in Canada. He later joined New Mexico's Los Alamos National Laboratory, where he worked on a DARPA research grant until 2001. Since leaving Los Alamos, Tilden has been working with WowWee Toys to develop the technology into toys such as the Bio-Bug series and the recently-released RoboSapien.

As with Linux, the development of BEAM technology has as much to do with the community as it does the founder. There has been a surge of interest and a world wide community of support for BEAM robotics in the last decade, with new ideas and circuits coming out regularly.

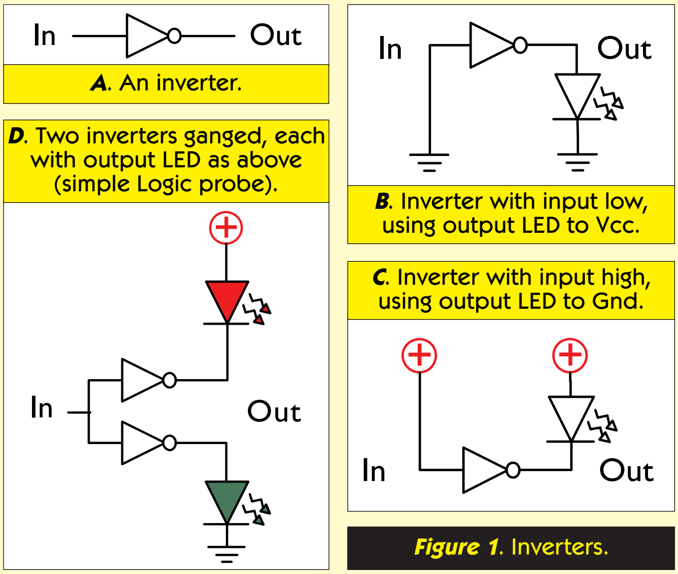


Figure 1. Inverters.

"low" or vice versa. The little circle at the point means, "input signal gets inverted here at the output." Logic "high" generally means that the input connects to the battery's positive terminal, while "low" means the input connects to the battery negative terminal.

If you're a relative beginner, we suggest you breadboard the circuits shown in Figure 1.

All is well and good when the input is clearly high or low, positive or negative, on or off, but what happens when the voltage is a little high or a little low? It turns out that the CMOS inverter has a specific voltage threshold where low turns to high and vice versa. The input switching point for the 74ACT240 and 74HCT240 is fixed at 1.6 V. The 74AC240 and 74HC240 are a bit more interesting because the switching point is half of the power supply.

Figure 2A. Bare Bones Photovore photos (one with GM, one with typical BEAM shaft-wheels).

Step 2: A Simple Robot

Why is this more interesting? With some creative misuse — by keeping the input voltage close to the switching point — Grant McKee of Solarbotics, Ltd., designed a simple light-seeking or line following robot (Figure 2). Two reverse-biased photodiodes form a voltage divider at roughly Vcc/2. More light on one photodiode raises the voltage on the inverter's input, causing the inverter to switch. The robot turns toward the light until it overshoots and there is more light on the other photodiode, lowering the voltage and causing the inverter to switch back. This repeating pattern gives a positive phototropism or light-seeking behavior and a waggling gait, suggestive of insect movement.

Grant called his robot the Bare Bones Photovore or BBPV. (In BEAM terminology, following the pattern of Herbivore/Carnivore/Omnivore, a Photovore is a critter that "eats" light. Occasionally, it is misapplied to any robot that

Parts List (for the series)

Electronics		
74AC240*	Texas Instruments SN74AC240N Fairchild MM74AC240N	Digi-Key 296-4305-5-ND Solarbotics 74AC240
Photodiodes*	Siemens SFH 205f Wide Field Salvaged from computer mouse	Solarbotics IR1
Assorted LEDs*	Tiny Red LEDs Tiny Green LEDs Tiny Red, Green, or Yellow LEDs	Digi-Key 350-1347-1-ND Digi-Key 350-1348-1-ND Solarbotics TLED
Resistors	Assorted, a few 470 Ω-1K * And a few in the 1M–10M range	
Capacitors	Assorted, 0.1 µF to 0.22 µF range	
Mechanics		
Motors*	Two matched hobby gear motors with wheels Two recycled matched "pancake" motors	Solarbotics GM6 with RW2 CD players or CD-ROM
5 V Power	4-AAA or 4-AA Battery Pack*	RS 27-411 includes switch
Power Switch	SPDT or SPST Power Switch*	Various
Misc.		
Breadboard*	Generic Solderless Breadboard and Ties	

* *Required for this month's projects*

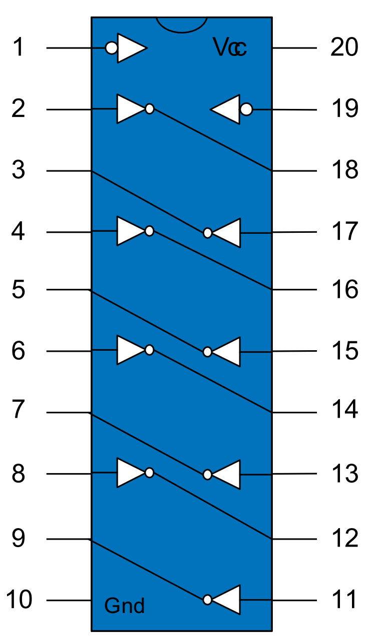


Figure 3. 74AC240 pinout.

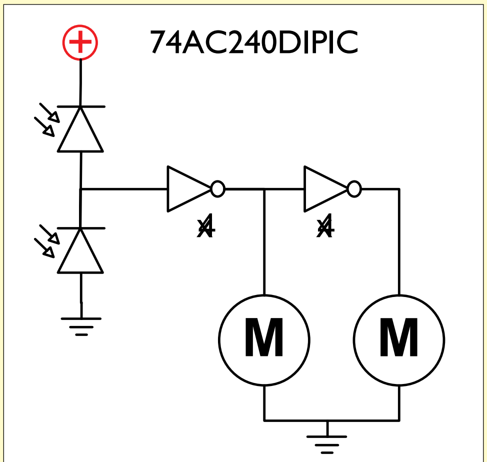


Figure 2B. Bare Bones Photovore schematic.

shows light-seeking behavior.) The BBPV qualifies as a BEAM robot because of its minimal part count and simple circuit and it is an excellent choice for your first BEAM project or for a parent and child project.

We will build the BBPV on the breadboard as a starting point. Take the 74AC240 and plug it into your breadboard. Be sure the notch is away from you — at the top as you look at it on the board. Figure 3 shows a 74AC240. Note the notch.

The 74xx240 chip has two halves, each with four inverters for a total of eight. The pins labeled "OE" with a line over the letters are Output Enables. The line on top tells you that they must be low to be turned on, so you must connect them to ground (battery V-) if you want the inverters to work. Connecting an OE to high (battery V+) will tristate or disable or turn off the four inverters it controls, which may sometimes be handy.

For example, a walker circuit might disable the leg motors so they don't flail about while the brain circuit is being stabilized. On the other hand, you might arrange to make the robot turn by disabling the motor on one side while the other keeps running.

Study the pinout in Figure 3. The inverter inputs are labeled 1A1, 1A2, 1A3, and 1A4 down the left, which is bank 1; up the right side, inputs 2A1, 2A2, 2A3, and 2A4 are bank 2. You can figure out the outputs. Take a pencil and sketch in each inverter from its input to its output, just so you see the pattern. It's important that you can accurately identify each input and output and match them up.

Before we start, here is a simple rule: Before adding or removing parts, connecting or disconnecting parts, always disconnect the battery first. This is the easiest way to avoid burning something out!

The photo (Figure 4) shows pin 20 connected to Vcc, which is the positive side of the battery and has a red stripe on our breadboard. Pin 10 is connected to GND or ground, which is the negative side of the battery and has a blue stripe on the board. Pin 1 is also connected to ground, so that bank 1 is turned on. (If your breadboard just has one power strip down each side, you'll have to make adjustments.)

Look at the photograph and the wiring diagram (Figure 5) to complete the circuit.

Once you have the circuit breadboarded, check out how several of the inverters wire together to power the motors. In BEAM tradition, we misuse the inverters as buffers or motor drivers. This is convenient because they are extras in our circuit, keeping the part count and cost down.

At first glance, this might look like a waste of inverters. You're welcome to try running the motor from a single inverter, though you're apt to be disappointed and might even burn out the IC. Most motors draw more current than the single inverter can provide, so either the motor will move like a rock or all the magic smoke will escape from your 74xx240 — or both.

You'll need to know two pieces of information to determine how many inverters or buffers you need to drive a motor: supply and demand. For example, a Fairchild 74AC240 can supply up to 50 mA of current per inverter. A

WELCOME TO BEAM!

For a good introduction to BEAM Robotics:
http://encyclobeamia.solarbotics.net/articles/beam.html

For a wealth of general information about BEAM:
www.solarbotics.net
Some of the information here is basic and more is advanced and technical, so you'll keep coming back again and again as your ability grows.

For a background of the skills and knowledge you need, go to the BEAM library at:
www.solarbotics.net/library.html
and check out "BEAM From the Ground Up."

More Information on the BBPV

For more information on the Bare Bones Photovore — including instructions on how to build the robot and make it solar-powered — check its website at **http://grant.solarbotics.net/Circuits.htm** or download documentation from **http://downloads.solarbotics.net/PDF/Bicore_Experimenters_PCB/BEP_App04-BBPV.pdf**

typical hobby motor can demand around 120 mA of current under a moderate load. Therefore, you need at least three inverters chained together (50 x 3 = 150, and 150 > 120).

Watch out, though, because motors draw much more current if they become stalled. The motor that demands 120 mA while driving your robot around could spike up to 650 mA if it gets stuck. Some BEAMers will piggy-back driver ICs on top of each other for high current requirements. Others may simply use a DIP socket and be prepared to replace the 74AC240 if it fries.

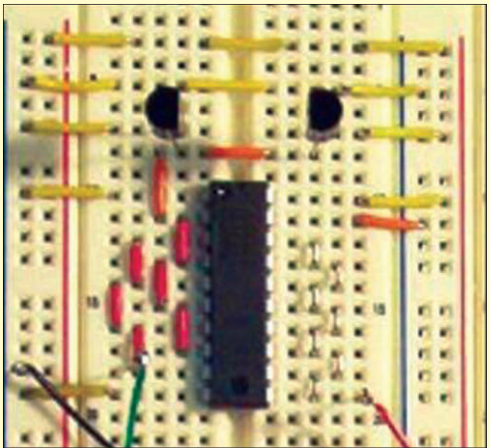


Figure 4. Bare Bones Photovore breadboarded.

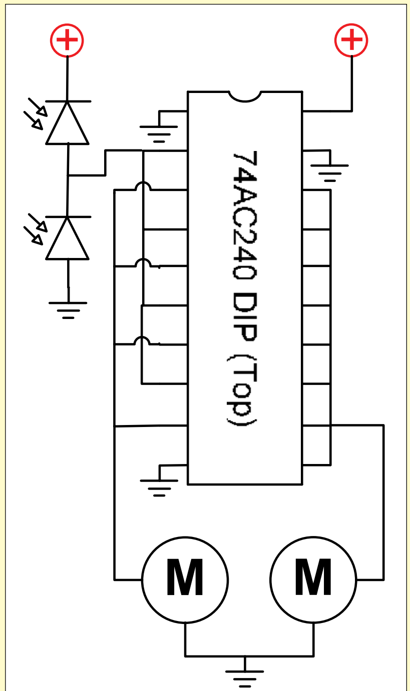


Figure 5. Bare Bones Photovore wiring diagram.

Summary

In this article, we have given a quick introduction to BEAM (Biology, Electronics, Aesthetics, and Mechanics) and presented a typical BEAM robot — the BBPV.

Next month, we'll explore the basics of Tilden's electronic neuron — the Nv — and give a couple of simple projects based on it.

Until then, you have plenty of time to take the BBPV off your breadboard and assemble it into a little critter on wheels. Figure 5 gives a suggested wiring diagram; the motors can simply be hot-glued onto the battery holder.

If you want to extend the project, there is a solar version with dark-activated battery power at **http://groups.yahoo.com/group/beam/** (You'll have to join the list to access the files and look up BBPV3_AL.gif) You can also have a try at pointing the photodiodes downward, turning it into a simple line follower.

Even in its simplest form, presented here, the BBPV will follow sunbeams across your kitchen floor or chase a flashlight beam and it will certainly intrigue your cat (if you have a cat) and your kids (if you have them). We hope it intrigues you a little, too! **SV**

About the Authors

*Tom is R & D Communications Officer for a company that manufactures hot tubs. Wolfgang works in IT and specializes in thin-computing solutions. Hobbyists with absolutely no formal technical or electronics background whatever, the two connected online at **http://groups.yahoo.com/group/beam/** and have never met face-to-face.*

What the (R.{6}\sE\w+s*\w{3}) is a Regular Expression?

by Jack Buffington

If you haven't worked with regular expressions previously, the third word of this article's title may look like a random grouping of characters. Hopefully, within the next couple of pages, you will begin to understand what they really mean. Regular expressions are all about searching for specific patterns of characters and conditionally replacing them within strings. Regular expressions can also be called *regex* and can be used in many different programming environments. Regex patterns can appear to be pretty complex at first, but they can be broken down into simple sub-statements that are easy to understand.

Let's start with a basic pattern. This is where the pattern that you are searching for is the same as the characters in the regex pattern. For this example, we'll use "tea." Notice that it matched every instance of tea, even if it was contained within another word.

Mike steals my tea
sometimes.

Figure 1. Matches for tea.

Let's look at another example. Regex has a pattern, which is **\w**. This will match any alphanumeric character or an underscore. If we applied this to the text in Figure 1, it would match every letter, but not the spaces. Regex allows you to create larger patterns out of smaller ones. For example, if we want to search for four letter sequences that start with "bar," we can use the pattern **bar\w**.

There is a
barnacle on the
barbell that is
sitting on the bar.

Figure 2. Matches for **bar\w**.

Regex allows you to specify a pattern that will match any of the given characters by placing them within braces. For example, **[Rr]** will match either a lower case or upper case R.

Red leather
Yellow leather

Figure 3. Matches for **[Rr]**.

Putting multiple characters within the braces will match any one of them, but not multiple letters from that group. For example **[dog]** will not actually match the word dog as a single unit, but will instead match the individual letters d,o, and g.

dog
Doug
drudge
good

Figure 4. Matches for **[dog]**.

Adding a **+** after any single regex pattern will match one or more characters of that pattern. For example, if we wanted to find any instance of a word that started with a capital R, we could use the pattern **R\w+**.

Raymond flew the
Jolly Roger flag
on the flagpole by
his house.

Figure 5. Matches for **R\w+**.

What if we wanted to find every word that started with a capital letter? To find those words, we could search using the pattern **[A-Z]\w+**. Some explanation is needed for the **[A-Z]** section. Regex (and computers, in general) have no idea what a capital letter is. In the pattern **[A-Z]**, don't confuse it with searching for capital letters. What it is actually doing is searching for any character that has an ASCII character code between A (65) and Z (90). Using the pattern **[M-m]** is perfectly valid and would search for characters with ASCII codes between M (77) and m (109), which includes the capital letters M through Z, [, \,], ^, -, `, and the lower case letters a through m.

You need unique
New York.

Figure 6. Matches for **[A-Z]\w+**.

Here is another example. Let's say that we are looking for any word that contains the letter A. An initial guess might

be to use the pattern **\w+[Aa]\w+**, but that won't actually match all of the words that contain an A. Instead, it will match words that have an A somewhere in the middle of the word. What we need is a modifier to replace the **+** that matches zero or more of a given pattern.

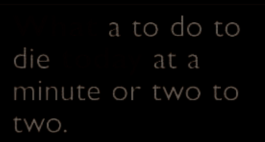


Figure 7. Matches for \w+[Aa]\w+.

The modifier that will match zero or more instances of a pattern is *. This makes our pattern **\w*[Aa]\w***. The pattern is starting to look complex, but it really isn't when you break it down.

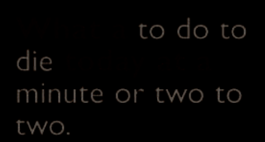


Figure 8. Matches for \w[Aa]\w*.*

It is possible to search for whitespace characters. These are characters that do not print, such as a space, tab, carriage return, line feed, vertical tab, or form feed. We can do that with **\s**.

Searching for letter and number patterns is something that regex does well. Let's say that you wanted to search a list for part numbers in some company's inventory that contain a letter, several numbers, and then two letters. You might make a regex pattern that looked like this: **\w\d+\w\w**. In this example, we use a **\d**, which represents a digit. This pattern would match the list in Figure 9 almost correctly.

Figure 9. \w\d+\w\w.

Although the regex expression did, in fact, find the pattern that was specified, it found subsections of part numbers, which is not acceptable. We need to find a way to find only complete part numbers. An initial thought would be to search for a whitespace character on either side of the part number. This is almost correct. If we did that, then the result would include the whitespace characters. Since all we want are the part numbers, we need to use another pattern. This pattern is **\b**. The **\b** pattern specifies a word boundary. It does not return a character in the match.

Part numbers can be dreadfully dull, but they are good for explaining regular expressions. Later in the article, there will be a description of how regular expressions are used in other ways than finding part number matches!

Figure 10. \b\w\d+\w\w\b.

There is an intentional error in most of the regex patterns described previously. Did you catch it? For example, although the pattern in Figure 10 has correctly found the valid part numbers in our previous example, it will incorrectly match a part number such as "346sd." This is because the pattern used **\w**. If you look back at the beginning of this article, you will see that **\w** will match any alphanumeric character or an underscore. To be perfectly correct, our example pattern would have to change to **b[a-z]\d+[a-z][a-z]\b**. If your part numbers sometimes include capital letters, then you would need to change **[a-z]** to **[A-Za-z]**. Of course, if you know ahead of time that this will never happen, then using **\w** is perfectly acceptable and will save you a little bit of typing.

A similar pattern to \w is the period. A period will match any printable or nonprintable character.

Take a look at Figure 11 for an example. You'll notice that Bob and kick have some letters selected. If we were to assume that there is a line feed character and a carriage return character between lines, then this would be a perfectly valid match. You need to be careful when you are writing regex patterns because, sometimes, you will get unintended matches. In this case, it is an easy fix to put a /b before or after the pattern to prevent this sort of match.

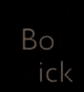


Figure 11. Matches for \b.

Let's say that you wanted to match part numbers that contained three letters, six numbers, and four more letters. That would make for quite a lengthy regex pattern: **\b[a-z][a-z][a-z]\d\d\d\d\d\d[a-z][a-z][a-z][a-z]\b**. — Yikes! Fortunately, regex has a solution to that. This solution is to use curly braces. For example, to match the six digits, we can

use the pattern **\d{6}**. Curly braces are a modifier that can be used after any regex pattern. This simplifies our lengthy pattern into **\b[a-z]{3}\d{6}[a-z]{4}\b**. Sometimes, you won't know ahead of time exactly how many characters will be in a certain pattern, but you have a good idea. The curly braces allow you to define a range for the number of patterns that you wish to match. Figure 12 shows such an example.

104
76006
500000
600,000

Figure 12. Matches for **0{2-4}**.

Let's suppose that your part numbers follow the following format: a letter, a number, a letter, a number, a letter, then a number. You can define a pattern for this by using parentheses. This pattern would be **([a-z]\d){3}**. Parentheses create a group of patterns that can be referenced as a single unit.

Let's get back to braces again and show another use for them. With braces, you can specify a pattern to *not* match. For example, **[^A-Z]** will match anything other than an upper case letter. An example of this is shown in Figure 13.

Hello
Test
test
Lollipop
green

Figure 13. Matches for **[^a-z]\w***.

Every implementation of regular expressions is a little different. In these examples, the whitespace characters have been largely ignored. For example, in Figure 13, the carriage return characters between lines would likely be matched by the **[^a-z]** in the implementation of regex that you would use. A more accurate pattern for Figure 13 would be **[^a-z \s]\w***. This would match sequences that do not start with a lower case letter or a whitespace character and have alphanumeric or underscore characters for the rest of the sequence.

Regex patterns can become more sophisticated than what has been shown here. Using regex patterns, you can match pretty much any pattern that you can come up with. Although it hasn't been discussed here, regular expressions can be used to replace the selected items with other items or to reorder the found items. This can be every bit as useful as simply finding specific patterns.

Not all implementations of regular expressions are exactly the same. The differences between various implementations of regular expressions are minor, but they do exist. You will want to read any documentation provided with your compiler to fully understand how to use its version of regular expressions. There is a regular expression tester online at **www.roblocher.com/technotes/regexp.aspx** that operates similarly to what has been described here, except that it only matches the first instance of the provided pattern.

There are many programming languages that support regular expressions. Some of the more popular languages are Perl, JavaScript, Sun's Java, and Microsoft's .NET language.

Obviously, regular expressions are not something that is commonly done on small microprocessors that are found in a lot of smaller robots these days — with the possible exception of some of the Java-based Stamps. Regular expressions can be used in larger robots that have room for a processor that has more horsepower and RAM. It can be used in these robots to help verify user input or to search large databases for specific data.

Regular expressions are often used to verify information entered into Perl applications that are on the web because some clever people have figured out how to enter information into web forms that will cause the Perl program to crash and allow them to take control of the server running the program. Regular expressions can be very useful to reorder information in a database.

For example, if you have a database that has the name, address, phone number, and birthday of 1,000 people and you would like to reorder and add commas between each item, regular expressions can make this process a breeze.

Knowing how to use regular expressions can be a handy skill to have. Hopefully, this article has helped you understand how they might be used and has shown you how to do some simple searches of your own. There are several books available, as well as many sources of information on the Internet about regular expressions. These will be able to give you a more in-depth understanding of regular expressions and how to write them.

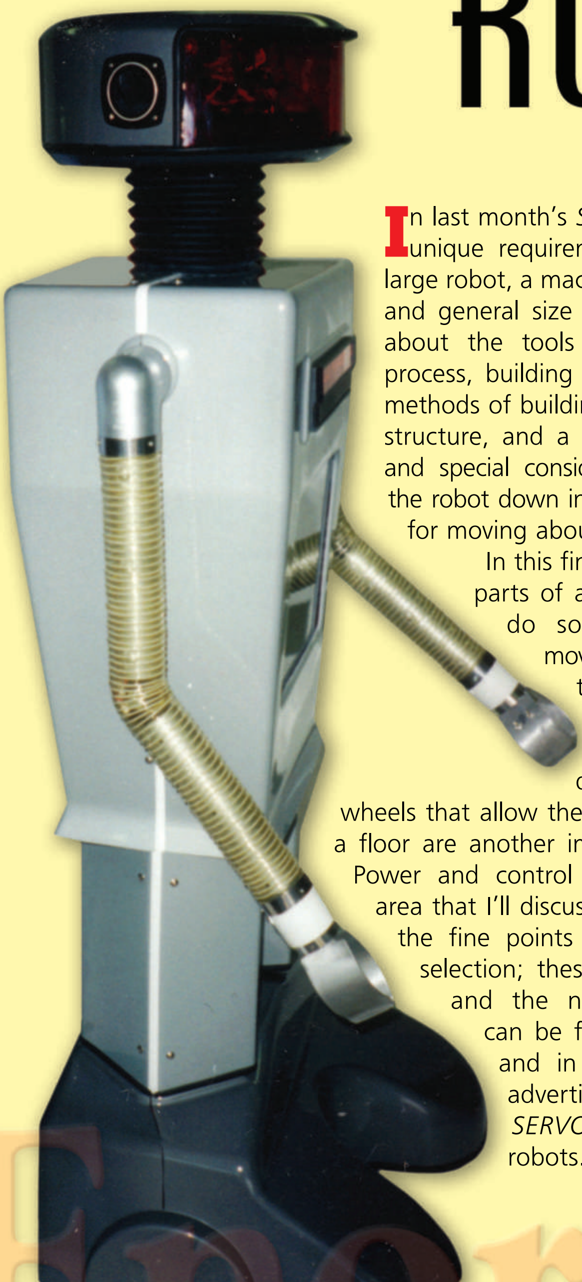
This column focuses on algorithms and data structures. So far, this column has sat firmly in an ivory tower and abstracted things. You can look forward to the next few months where, I will present some topics that can be immediately applied to smaller robotic projects. **SV**

Author Bio

When not writing for *SERVO Magazine*, Jack runs Buffington Effects, a company that designs and builds animatronics and motion control devices for the entertainment industry. Check out his website at **www.BuffingtonFX.com**

Building a LARGER ROBOT Part 2

by Tom Carroll



In last month's *SERVO*, I discussed the unique requirements of designing a large robot, a machine with the features and general size of a human. I talked about the tools needed, the design process, building shell pieces, different methods of building the robot's internal structure, and a bit about body joints and special considerations for breaking the robot down into convenient sections for moving about in a small car.

In this final part, I'll discuss the parts of a robot that physically do something — that is, move. The arms are usually the additions to a large robot which most builders desire. The drive systems or wheels that allow the robot to move about a floor are another important design area. Power and control systems are another area that I'll discuss, but I won't go into the fine points of motor or battery selection; these are speciality areas and the necessary information can be found on the Internet and in many of the books advertised in *Nuts & Volts* or *SERVO* about combat robots. You should read all you can about motor design and control before you purchase your motors — especially the main drive motors. A little advance knowledge will save you a lot of grief over improperly selected drive and control components.

Robot Arms

Arms are always a major desire where large robots are concerned, especially robots that resemble human arms. It seems when we graduate from a small tabletop machine to a larger robot; the addition of one or two arms is at the top of our wish list.

Needless to say, the more axes of motion or degrees of freedom that you use, the more complex the mechanisms and motors required. I used single axes of motion at the shoulders on the dentist's robot to cut costs. The arms that I used for this robot seemed a bit too skinny to me but the customer liked them. I added some flashing colored LEDs inside the transparent arms that looked cool, along with brightly colored wires. The elbows could be bent to a desired angle and would stay in that position with a friction joint, but you most likely will want to add another axis of motion at the elbows for your robot.

Powering the motion of a robot's arms

always seems to be a problem with builders. Using a "brute-force" gear motor at the shoulder joint seems to solve the requirements of most builders, but adding motion to other joints causes many first-time builders a lot of grief. Motors are easy to hide inside the robot's chest cavity, but skinny arms are difficult to accommodate elbow and hand actuators. The use of flexible rotating shafts running from motors within the chest to arm joints is one very good way to eliminate heavy motors in the arms. Jim Hill used this method in his robot, Charlie. You can use small gearboxes at each joint or linear actuators that act like our arm muscles Jim used.

In the four robots that I built for *Revenge of the Nerds,* I used another elbow motion system that combined a single actuator (the shoulder motor) to actuate both the shoulder and elbow. I used a "figure eight" cable attached to a fixed pulley at the shoulder (Figure 2). When the upper arm was moved in an upward motion and the shoulder pulley stayed still, the cable would tug on a pulley in the elbow to cause the upper arm segment to move the same amount of degrees.

Thus, when the upper segment moved 45°, the lower arm would also move 45° and the arm would end up facing 90° straight forward — a very normal arm movement for a human. All this was accomplished with only the single shoulder motor.

Arm Mass Compensation

Another thing that you will discover when you decide to place arms on your robot is just how much force is required to move a weight up 90° with the arm. If you measure your robot's arm weight, you might find that it weighs, for example, two pounds at the end of the hand or claw. That may not sound like a lot until you calculate the motor torque required just to lift the arm. If the arm is two feet long, that's four foot-pounds of force — or 768 ounce inches of torque — required just to lift the arm with no payload. Holy cow! That is going to require a big gear motor.

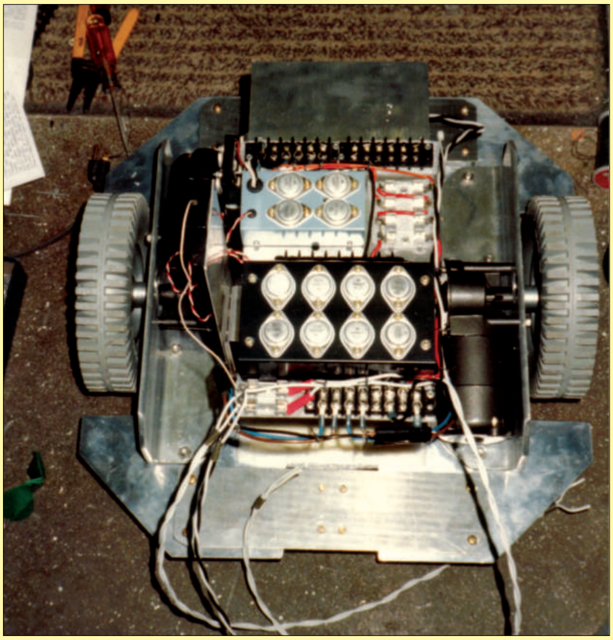


Figure 1. Peeking inside the robot base.

Don't despair. There are two ways to make a large arm lift a sizeable payload. One method is to use a spring to compensate for the arm's weight. You can use a large coil spring around the shoulder joint, but a better way is to use a gas spring on a short lever arm inside the robot. Gas springs are used on SUV tailgates and car trunk lids; they are fairly linear force compression "springs" and can be found at surplus places or auto parts places. Having the gas spring push against, say, a 6 inch lever inside the robot's body would help lift the arm. With a force of 24"/6" or four times the weight at the end of the arm (2 pounds), a spring that had a force of 4 x 2 pounds would render the arm "weightless," allowing all the torque developed to go to lifting a payload.

A 4 inch lever and a spring pressure of 12 pounds would also create the same effect, but think a minute. Why just take away the arm's empty weight? Why not compensate for a payload? Suppose you found a gear motor or linear actuator that you wanted to use that can easily create 20 foot pounds of torque at the arm's shoulder joint and you'd like to lift at least 10 pounds. If you use a 48 pound forced gas spring on a 6 inch (4:1 ratio) lever arm (48/4 =12 lbs lift — 10 lbs payload + 2 lbs arm weight), you can compensate for the two pound arm weight and a 10 pound payload.

Needless to say, you're probably going to have to live with the gas springs that you can find. Instead of trying to locate a gas spring that has the exact force that you want, change the length of the small lever arm to arrive at the weight compensation that you desire with your spring. With the payload and arm now compensated for, you can lift 20 pounds of payload with the same actuator. Of course, with the arm's claw empty, the actuator now has to pull the arm downward, which it can easily do with the actuator you selected.

Making the Robot's Arms Move

Moving the arm can be accomplished in several ways. You can directly connect a large gear motor with enough torque and low enough speed. Think about the arm's actuation speed when you select any sort of gearmotor or actuator system. Say, for

instance, you find a nice, beefy gearmotor that has 10 rpm under load — one revolution every six seconds. If you want to use it for an arm motor, realize that the motor will turn 1/4 revolution (a typical arm's movement) in 1.5 seconds — pretty fast.

You can even use a direct drive gear motor at the shoulder, but this is going to be a very hefty motor — even with gas spring weight compensation. I once used a pair of off-road winches for the arms — stripped to just the motor and gearbox with shaft — on a robot that I built. There are some amazingly cheap winches available from Harbor Freight Tools, but check the specs carefully or look at the winches at one of their stores. "Quality" and "guarantees" are not a strong point of some of the import items from China.

Another method is to use radial rods on the shoulder joint inside the robot and the same length radial rods on a drive gearmotor below the shoulder. Two push-pull metal rods connected between the two can cause a 90-110° rotation, sufficient for robot arm rotation.

Remember to use limit switches for protection. The same goes for using a DC winch motor. A 2,000 pound winch with a 2 inch drum diameter (1" radius) turning at 3 rpm can lift 83.3 pounds (2,000/24") 90° in 5 seconds (60 seconds / 3 = 20; 20 seconds / 4 = 5 seconds). This is a nice lifting force at a good speed, but remember to use some pillow block or flange-mounted bearings for the shoulder shaft to handle the forces developed.

You can also use a DC electric linear actuator to pull or push on an internal lever arm in the same way discussed earlier for the gas spring lever. These actuators are affordable and quite powerful for their size. Many of the large satellite dish steering actuators are 115 VAC, but many 12 or 24 VDC actuators can be found at surplus houses.

Take my advice and use limit switches for protection. The use of chains around drive and driven gears is also popular. However, in your design, you will probably have a fairly large gear to deal with at the shoulder of your robot. The same goes with a toothed, belt driven system, so you may want to use a partial "pie shaped" segment gear.

Figure 2. Shoulders with cable arrangement.

SCARA Robot Arms

Another arm configuration worth mentioning is the SCARA (Selectively Compliant Articulated Robot Arm or Selective Compliance Assembly Robot Arm) arm that is probably the most common industrial robot configuration. The SCARA robot is extremely popular in electronic and other precision assembly tasks. Instead of the more common horizontal axis arms that bend up and down, the SCARA arm has vertical axes that allow the arm to swing side to side in a semicircle. Figure 3 shows a British design I worked on a while back called the R-Theta by Universal Machine Intelligence — unfortunately, they are no longer in existence. It looked great, but the arm was a bit flimsy. It is a SCARA arm mounted on a motorized mobile base.

SCARA arms aren't well adapted to reach down into holes or even to the floor to grasp objects, but they can easily maneuver and lift an amazing amount of mass for their size. You can easily move a 100 pound door with your fingertip because the hinges are all vertical — just as in a SCARA arm. The arm's motors only have to overcome the payload's inertial mass, not its gravitational weight.

The use of a leadscrew assembly to lift the arm at its shoulder can allow a large weight to be lifted and moved around in a semicircle. Acme — or better still — recirculating ball-screw leadscrew assemblies that can be easily adapted to DC motor drives for lifting SCARA robot arms are available at many surplus stores and Internet dealers. I saw four D Ni-Cad cells power a small 10 inch ball screw/motor actuator that had a force of over 200 pounds.

The Drive Motors for the Wheels

Arms are great for maneuvering objects, but your main drive system is what will allow your robot to roam about on your command. The selection of drive motors and wheel systems is one of the most important considerations in the design of a large robot. I had long been familiar with the potential uses of electric wheelchair systems; a robot I was building and a person were about the same weight. I chose the A-BEC motor/wheel assemblies that I had used many times before. They are quiet, powerful, and easy to mount. The attached wheel could support several hundred pounds of force directly on the output shaft.

The very best part about using these motor/wheel assemblies is that you don't have to worry about calculating wheel forces on your robot's bearings. These units are made to support the weight of a human and more; they are certainly enough for a large robot. These types of motorized wheels also have a hub that can be unlocked for

freewheeling, in case someone wants to push the robot along without it being powered. At 24 volts, the motors only drew about 20 amps in a "rotor-locked" or stalled condition.

Another similar wheelchair gear motor has been making the rounds of the surplus places; it's an import from Huafeng Electrical of China. They were designed for wheelchair use and are ball-bearing units. They do not have the wheel assembly attached, but can be purchased as a matching pair — right and left mounted — for about $300.00 a set. Rated at 24 VDC and 120 inch-pounds at 94 rpm, they seem to work well with a belt or roller chain drive system. C & H Sales in Pasadena, CA (**www.candhsales.com**) has the units in stock — part number DCGM2103RH for the right hand drive and DCGM2103PR for the left.

Steering Configurations: Differential or Ackerman

This is a good point to discuss the two main types of steering configurations before you do your final selection of drive systems. The motor/wheel combinations are best for the most common type of robot drive configuration — the differential or "tank type" system. With this configuration, the robot steers to the left by increasing the speed of the right wheel (or decreasing the speed of the left wheel) to make a turn, just like a military tank. The greater the difference between the two wheel speeds, the sharper the turns to the point. If both wheels are turning in opposite directions, the robot will spin on its axis.

This configuration requires one or more passive, freewheeling swivel casters at the front, back, or both locations to stabilize the robot. With the mention of "tank type" steering, you might be tempted to use treads instead of wheels. They may look cool, but my advice is not to because they must skid when turning, making them inefficient for battery-powered robots.

Also, they can make a mess of carpeting, floors, and even grass. Go for wheels, instead.

One very important point to mention is that the casters used to stabilize your robot must be mounted in a way to allow them to spring up and down on uneven surfaces so that the main drive wheels will not become high-sided when they drop into a slight depression. The spring force on the casters must not be so strong that the weight of the robot will not push the main drive wheels down into a depression.

Conversely, they must not be sprung so lightly that the robot will bob when it is quickly powered and then stopped. The old 20 pound Androbot TOPO used two canted drive wheels and two small casters; it bobbed back and forth like a child's toy — not good for a 200 pound, human-sized 'bot.

The other type of drive system is the Ackerman or "car type" of configuration that has one or two steering wheels at the front (or even back) of the robot's chassis. Beside the cars that we drive, model R/C racecars also use this type of system and many robots are made from these cars. They do not have the capability of turning on an axis like the differential configuration, but they do have the capacity of traveling straight forward with no difficulty.

If the wheels are pointed straight, the robot will pretty much travel in a line. It is when the robot must turn that this type of system runs into a bit of difficulty and that is the reason that most robot builders use the differential configuration.

Take a look at some of the configurations of motorized wheelchairs and electric scooters. These units are designed to carry several hundred pounds for miles at walking speeds — just about what a large robot might be expected to do. The wheelchair that I mentioned earlier is great for turning in a tight place, as are some of the other units that have two side wheels and swivel casters. Other scooters use two passive back wheels and a single powered front wheel for steering. It may be worth your effort to use an old scooter or wheelchair base as the base for your robot.

Delivering Power to the Wheels

Chain or belt drives are another

Figure 3. Robot with SCARA arm.

way to bring the power of a nice pair of gear motors you've found to your robot's wheels — like the Chinese wheelchair motors mentioned earlier. Using the flexible chains or belts allows a bit of slop in the distance between your wheels and the drive motor. You can select your own gear ratio to compensate for any speed problems you may have with your selected gear motor output speed and wheel diameter.

Use the gearmotor's highest output speed to determine the ratio that you'll need to obtain your robot's top speed. A speed of 2-3 mph is fine for a teleoperated machine with you in the loop for visual feedback, but 1 mph might be top speed for an autonomous bot with all the sensor fusion and microcontroller processing involved.

Applying power from your gear motors to the wheels can be accomplished in many ways, but I'll discuss the two most popular methods. The most used method is to mount the wheels on a stationary shaft. You need to use wheels with a pair of built-in bearings with inner diameters that fit snugly on your shaft. A pair of setscrew collets hold each wheel onto the shaft. A ring gear or pulley is mounted on the inside face of the wheel with the center cut out so the shaft/axle will pass through.

A drive chain or toothed belt outside the robot's base transfers power from your drive motor.

The other method is to use two rotating shafts fastened to the wheels — just like a car. Two sets of bearings hold the shaft in a horizontal position and a pair of drive gears are fastened to the shafts that receive power just like the free-wheel system mentioned above. This arrangement is a bit more complex mechanically, but allows all the drive belts/chains to be inside the base.

One critical thing to remember in the selection of drive motors for a large robot is the potential for very large current draw. This fact will certainly affect the design of your drive circuitry. A completed large robot may draw 10 amps when running around your garage floor during initial testing. In normal operation, the motors may draw 25 amps on a deep pile rug, 35 amps in your yard, upward of 70 amps on a slope, and maybe 100 amps in the "locked rotor" condition where the motor is stalled.

You cannot tell yourself that you won't have the robot near such obstacles, as it just might try to bump its way through a wall without you directing it to do so. An expensive motor controller or H-bridge might go up in smoke in this situation.

The A-BEC units were expensive, but very efficient. Of course, the current draw will depend on the motors used, their efficiency, the supply voltage, the weight of your robot, starting loads, and the surface the bot is operating upon.

The way I tested the drive systems on a large robot was to use what I called a "poor man's prony brake" made from a 1 x 2 inch stick, a piece of carpeting, a turnbuckle, and a 50 pound spring "fish scale." I'd tack the carpet strip to one end of the stick so that the robot's tire was in contact only with carpeting. I'd loop that around the tire and hook the turnbuckle to the free end of the carpeting and the other end to the stick. The "fish scale" was attached to a point on the free end of the stick two feet from the center of rotation.

When the wheels turned, the stick was forced downward in proportion to how tight the turnbuckle was. The tighter the turnbuckle, the tighter the carpet strap was around the tire, resulting in more force on the stick. So, if I measured 10 pounds of force pulling on the "fish scale" (minus the weight of the stick at that point), I had 10 x 2 feet or 20 foot-pounds of torque. Further tightening of the turnbuckle, I could maybe get 80 foot-pounds of torque before the motor stalled. This is a cheap and dirty version of the classic dynamometer used in car test facilities that can apply inertial — as well as frictional — loads to a vehicle's wheels.

In coming up with this crude setup, I tried rubber strips — which would suddenly bind on a rubber tire — and even a leather belt that was hard to keep on the tire's surface. Sometimes, the carpeting would work its way off the tire if I wasn't paying attention, especially when I was working with tires with curved surfaces. I found the process to be a bit easier if I had a friend hold the carpet/stick/scale arrangement and I controlled the motor current and adjusted the turnbuckle.

Obviously, the arrangement will need to be a bit different, depending on the tire size being tested. Just a word of advice — don't keep the motor overloaded or stalled too long, as the armature windings, commutator, and brushes may get permanently damaged from too much heat.

As I was testing the torque, an encoder read the rpm. I also used a current shunt to measure current draw on larger motors and a digital multimeter with a 20 amp scale for lower current motors. The particular shunt that I use is similar to a power resistor with two voltage taps on it. At the taps, I can read off 50 millivolts for every amp of current draw across the shunt, so a 1 volt output represented 20 amps of current draw across the shunt, in series with one of the motor's leads and so on.

With this simple setup, I could read speeds up to several hundred rpm,

100 amps of current draw, and 120 foot-pounds of torque. Knowing this data ahead of time certainly saves headaches later from undesireable speeds, an under-powered robot, or blown out driver circuitry.

Hey, if you don't want to go to all the trouble to make a poor man's prony brake, use a pair of heavy gloves to grasp the spinning wheel to simulate a load and read the current on a panel meter or multimeter in series with the motor.

Count the revolutions of the wheel in one minute and multiply by πD (π times the diameter of the wheel in inches) to find the inches traveled in one minute. You can feel the force with your hands to get an approximate torque reading. Strain gauges and dynamometers have also been used by robot builders with access to better instrumentation. Use your imagination.

Batteries — The Robot's Portable "Lunch"

Batteries are another very important consideration. A complete series of books can be written about batteries and still not cover all aspects. I met a promotional robot operator who had his robot tip over in the back of his van. The liquid acid in the automotive lead-acid battery spilled out and not only ate out the bottom of the robot, but also ate a hole in the floor of his van. Most likely, you will select a gelled-electrolyte or sealed lead-acid battery for safety reasons. Hawker Batteries, Power-Sonic, Panasonic, Carefree Magnum, and other brands are all fine for this type of application, as the severity of use does not match those used in combat robots. The Hawker batteries are preferred by many of the combat robot builders, as they can take quite a bit of load abuse, but they are a tad more expensive than the others.

Most batteries are rated in amp hours (AH) over a period of 20 hours. In short, a 20 amp hour rated battery can be discharged at a rate of 1 amp for 20 hours, a 60 amp hour battery at 3 amps for 20 hours, and so on. It is important to note that the 20 AH battery cannot be discharged at a rate of 20 amps for 1 hour or the 60 AH at 60 amps.

Quite often, combat robots demand a higher load from a battery than its rated load. Many participants use discharge specifications that are in a period as short as six minutes or less — a typical combat robot round's length.

For a short period like this, the amp hour rate is approximately 1/3 or even less the 20 hour discharge rate. For a typical large robot, you should consider only the 20 hour rate for your battery sizing calculations.

These few pages were not intended to be a class in "Large Robot Building 101," but rather to inspire you to go for building the larger machines. Maybe your tabletop maze robot has reached the limits of what you can accomplish with it or maybe you just want a servant that can take out the trash, put out the cat, and bring you a cool one from the fridge. One nice thing about larger machines is that you have a lot more room to mount sensors, computer systems, RF links, and many types of manipulator systems. Whatever the reason, I encourage you to try building a large robot. SV

Resources

Harbor Freight Tools
www.harborfreight.com

C & H Sales
www.candhsales.com

National Power Chair
www.npcrobotics.com

Our resident expert on all things robotic is merely an Email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by
Pete Miles

Q. I recently upgraded the sensors of my sumo robot to using two Sharp GP2Y0D02YK distance sensors, but the BASIC Stamp keeps resetting all the time. I tried changing the batteries from four AA batteries powering the robot to using the AA batteries for the motors and a 9 volt battery for the Stamp. The data sheet says that the maximum current is only 50 ma, but the resetting problem says I am drawing too much current from somewhere. Since this problem started when I added the Sharp sensors, I suspect they are my problem. Do you have any suggestions on how to get them to work?

— Jim Valentine
via Internet

A. The Sharp GP2Y0D02YK Long Distance Measuring Sensor is one of the better long range infrared object detectors. It has a fixed detection range of 80 cm and outputs a high signal when an object is within the detection range. This sensor has a very misleading specification — the draw. The specification calls out "average dissipation current" and gives typical and maximum values of 33 and 50 ma, respectively. What becomes misleading is that there is a maximum rating for an "average" current, leading you to believe that it is the maximum current draw.

This is not the case with this sensor. When they state average, they mean average. The problem you are having is the peak current draw from the sensor, which is causing your BASIC Stamp to reset. Since the data sheet does not provide a maximum current draw, I hooked up my oscilloscope to one of these sensors to measure the current. The current was measured by placing a 1.0 Ω resistor between the sensor and the power supply and measuring the voltage drop across the resistor. The current is then calculated by dividing the measured voltage by the resistor. Since I used a 1.0 Ω resistor, the measured voltage turns out to be equal to the current.

Figure 1. Current draw from a single Sharp GP2Y0D02YK distance sensor.

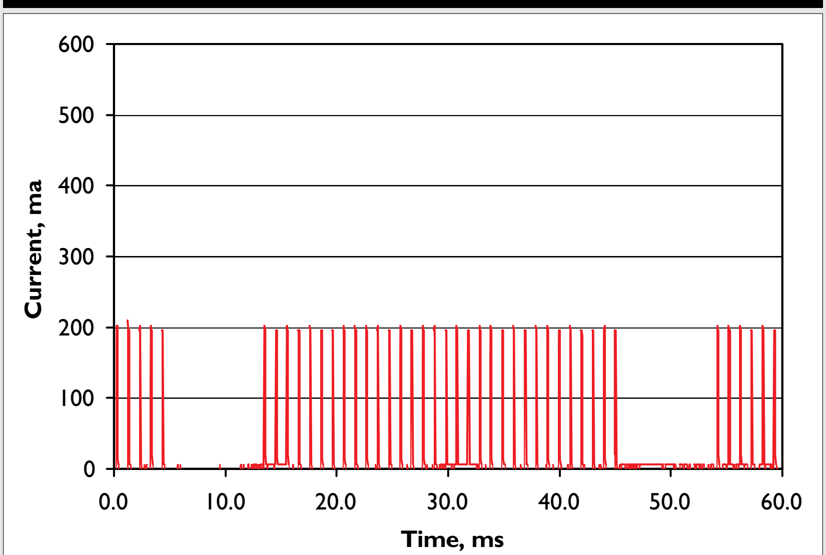


Figure 1 shows the results from these measurements. Here, you see a 31.7 ms long pulse train (12.5% duty cycle at 983 Hz) that consists of a total of 32 pulses. Then, almost no current draw for about 9.1 ms. What is surprising to see is that the peak current draw was about 200 ma, which is over four times the specified "average dissipation current" maximum. This shows that the peak current needs to be kept track of because it could cause momentary drops in the supply voltage.

To see how bad this can get, I hooked up three of these sensors to a BASIC Stamp 2 and used an external 7805 voltage regulator (a 5 volt regulator) to supply power to the whole system from a 9 volt battery. Figure 2 shows a small snapshot in time of the current draw with this test setup. Here, you can see that — at certain times — all three infrared LEDS are firing at the same time, which causes momentary current draws of over

half an amp, even though the average current draw in the system was 70 ma.

The original question was about trying to determine if these sensors were the cause of the Stamp resetting. To answer this question, an oscilloscope was hooked up to monitor the system voltage changes due to the current draw from the sensors and a time snapshot is shown in Figure 3. This figure shows that the supply voltage dropped down — for an instant — to 4.07 volts and then went back up to the normal 5 volt supply rating.

Is this a problem? Well, that depends on the microcontroller and support electronics and how well they work in low voltage situations. For example, the BASIC Stamp has a low voltage detection circuit (called a brown out circuit) that is designed to reset the Stamp when the supply voltage drops below 4.2 volts. In this demonstration, the BASIC Stamp did reset on occasion, due to the combined instantaneous current draw from these sensors.

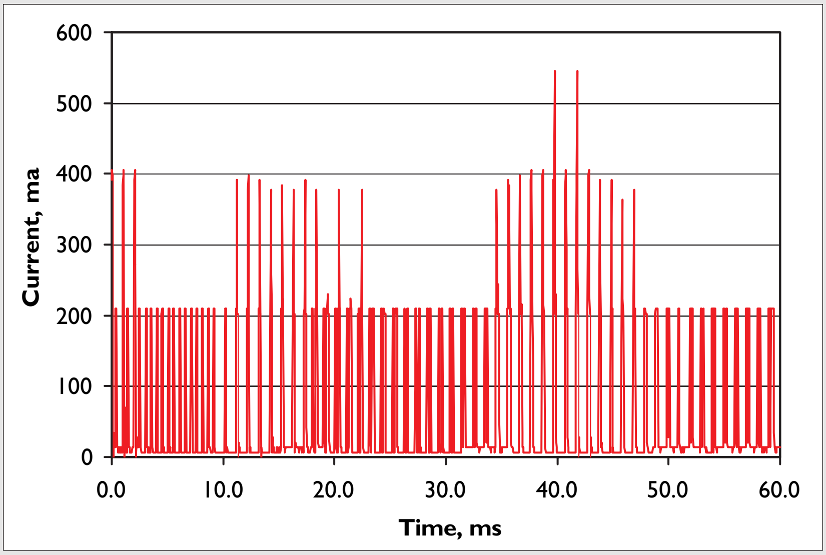


Figure 2. Current draw from three Sharp GP2Y0D02YK distance sensors.

Though the instantaneous current draw from these sensors is causing the BASIC Stamp to reset, this can be corrected by adding a 220 µF electrolytic capacitor across the +5 volt wire and ground wire near the sensor (see Figure 4). There should be at least one capacitor per sensor in the system; they should be placed as close to the sensor as possible. The 220 µF rating is just a recommendation. The larger, the better, but anything smaller than 100 µF doesn't provide enough help to bother with.

A final remark is that you will find the same type of results for both the Sharp analog and digital distance sensors — GP2D12, GP2Y0A02YK , GP2D15, GP2Y0D012YK.

Q. Are there any cheap programs that allow me to make my laptop act like an oscilloscope? I would like to be able to test my robot when I am not at school.

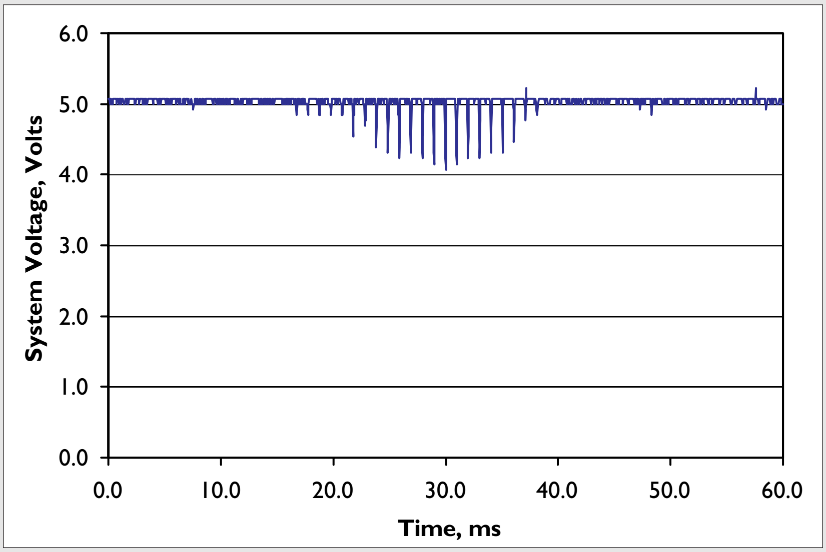
— Steve Anderson via Internet

A. What you need is a data acquisition system called a PC Oscilloscope, which is not just a program. Basically, a PC Oscilloscope system consists of two parts — hardware and software. The hardware, in essence, is an Analog to Digital (A/D) converter that takes the voltage measurement, converts it to a digital signal, and sends it to the PC via either the parallel or USB ports on your computer. The software then takes this data, converts, filters, and manipulates it before displaying the results in a window that has a similar functional appearance to a traditional, bench top oscilloscope. There is not a lot to it and you can make one yourself if you have a data acquisition board and LabView software from National Instruments (**www.ni.com**). However, that is the hard way to go about solving your problem.

There are quite a few companies around the world that make PC-based oscilloscopes with sampling rates that range from low end 20 kS/s (20,000 samples/second) systems to high end 5 GS/s systems. As a general rule, their prices go up as the sampling rate goes up (from $150.00 to over $1,000.00). Table 1 provides a short list of several companies that sell PC-based oscilloscopes.

Unless you are trying to measure the actual speed of a 20 MHz oscillator or measure radio frequencies, a high speed oscilloscope is generally not needed for about 99% of the robotics applications. I personally have the $189.00 OPTAscope 81M, sold by Parallax (**www.parallax.com**) and think it is probably the best buy for your dollar. It is a

Figure 3. Voltage drop due to the current draw from three Sharp GP2Y0D02YK sensors.



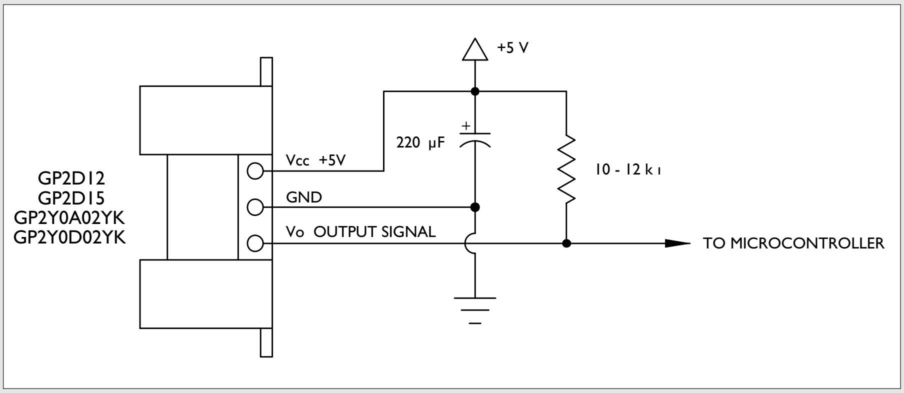


Figure 4. Schematic with a capacitor.

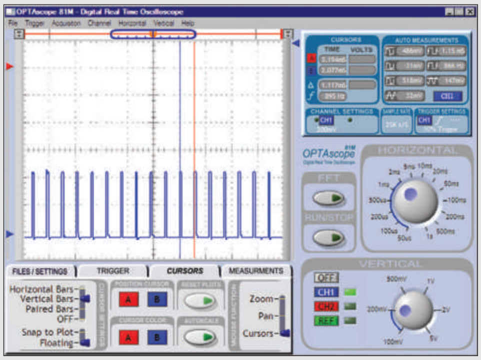


Figure 5. Screen shot of the Optascope, a PC-based oscilloscope from Parallax.

1.0 MS/s dual channel storage oscilloscope that connects to the USB port of the PC and includes three probes: two for measuring the two channels and one for external triggers. The power for the OPTAscope comes directly from the USB port, so no external power supplies are needed. This turns a laptop into a field-portable robotics diagnostic and development system.

Figure 5 shows an actual screen shot of the OPTAscope; it shows many of the features it has. As you can see, it has the same look and feel as a traditional bench top oscilloscope.

Since this is a storage scope, you can go back and take measurements of the signals with a set of movable cursors or save the results in ASCII text or Excel formats for additional offline analysis. All of the data shown in Figures 1 through 3 were obtained using the OPTAscope.

Searching the websites shown in Table 1 and using a keyword search of "USB Oscilloscope" should yield all the information you need to determine which oscilloscope to use with a PC. Just keep in mind what you plan to measure and use that to help you select which oscilloscope to obtain.

Q. I have heard that there is a remote shut off required in sumo, but I can't find the official rules that state that. Is there any truth to this? If so, how do you do this?

— Kathryn Lobb
Dickinson, TX

A. There were a few rule changes in Japan last year regarding the 3 kg weight division in robot sumo. They all relate to safety. First, all competitors must wear gloves and eye protection. Eye protection is always a good idea when there is a chance of eye injury due to flying debris. Some of the Japanese competitors literally sharpen the front edges of their scoops with wet stones. This was observed in Seattle, WA last March during the International Robot Sumo Tournament (IRST).

The Japanese use either sharp edges on their robots or flexible, very thin sheet metal edges that they replace after each match. Also, because the Japanese allow vacuum and magnetic systems on their robots, many robots will stall their motors during a match, creating a lot of heat. Because of this, gloves are now required for all of the competitors and their teammates when handling the robots. Gloves are also a good idea to use when handling any combat robot.

The last major change in the Japanese rules was requiring all autonomous robots to have a remote shut off. Some of the robots are lightning fast. They can traverse across the five-foot diameter ring in less than half a second. They then fly off the ring and across the floor. This can cause a serious safety problem for spectators, judges, and competitors — especially when trying to stop a high speed robot that has an extremely sharp front edge.

Right now, they don't have a specification on exactly how to remotely shut off the robot. The only thing they require is that the operator must be able to shut down the robot on command from the judge or be disqualified. For those of you who can read Japanese, these rules are outlined at **www.fsi.co.jp/sumo/index.html** The English version of the rules at **www.fsi.co.jp/sumo-e** has not been updated for over five years, so you won't find any reference to the rule changes there. As a side note, I am not aware of any sumo event in the US or Canada that requires adherence to these rule changes in their local events.

When the Japanese were visiting the US last year for the IRST event, I saw four robots that

Model	Sample Rate	Manufacturer	Website
DS2200C	200 kS/s	Easysync	www.usb-instruments.com
81M	1.0 MS/s	OPTAscope	www.optascope.com
BS300	40 MS/s	Bitscope Designs	www.bitscope.com
DSO-2102	100 MS/s	Link Instruments, Inc.	www.linkins.com
SDS 200A	100 MS/s	softDSP	www.softdsp.com
Various Models	From 20 kS/s to 5 GS/s	Pico Technology Ltd.	www.picotech.com

Table 1. Some PC Oscilloscope manufacturers.

used infrared systems to shut the robots off and one robot used a regular R/C transmitter — just like you see for model airplanes. In the robots that used the infrared systems, I noticed that they used the same IR receivers that are commonly used for IR object detection systems. In fact, they looked just like the Panasonic PNA4601M and the Sharp IS1U60 sensors. The sensors were placed on top of the robots, facing straight up.

The way they were used was that — when the match was over — the operator stood above the robot and pointed the IR transmitter straight down, pressed a button, and the robot stopped. In one case, the operator got frantic and pressed the button many times while waving his hands. I guess it didn't work the way he wanted it to. I personally do not view this approach as a safe method for shutting down a robot because it required the operator getting above the robot and transmitting a shut down signal. If the robot is running away, the operator has to chase the robot and — if the batteries failed in his transmitter — he wouldn't be able to shut down the robot.

Now, the person with the R/C transmitter did do a proper job for setting up a robot shut down safety system. On his transmitter, he tied a set of rubber bands to the neck strap hook that is on the center of the transmitter and to one of the sticks on his transmitter. This caused the stick to be pulled toward the center of the transmitter. When the robot was running, he used his thumb to pull the stick away from the center of the transmitter. When the match was over, he released the stick and the rubber bands pulled it back to the center.

He most likely had a small microcontroller that looked for a 2.0 ms pulse width (the pulse width when the stick was pulled away from the center of the transmitter). This would close a relay giving power to the rest of the robot. When the receiver did not receive this 2.0 ms pulse, it would open the relay and cut power to the robot. This is a brilliant, simple, and very effective idea.

I hope this answers your question. **SV**

Resources

Sharp — **www.sharpmeg.com**
Parallax — **www.parallax.com**
National Instruments — **www.ni.com**
Panasonic — **www.panasonic.com**
Easysync — **www.usb-instruments.com**
OPTAscope — **www.optascope.com**
Bitscope Designs — **www.bitscope.com**
Link Instruments, Inc. — **www.linkins.com**
softDSP — **www.softdsp.com**
Pico Technology Ltd. — **www.picotech.com**

DARPA Grand Challenge *Spawns* SRS/*SERVO Magazine* Robo-Magellan

by Michael Miller

I first heard of the DARPA Grand Challenge event back in 2002 — a whisper that the DoD (Department of Defense) was creating a "challenge" to foster growth in the development of autonomous vehicle technologies. In January of 2003, it was formally announced that there would be a fully autonomous robot race between Los Angeles, CA and Las Vegas, NV. A cash award of $1 million would be bestowed upon the first vehicle to complete the course on March 13, 2004.

The idea of designing and creating an entry excited me, but — after a few days of thought on the problems and solutions — the magnitude of taking on this sort of a project was beyond both my time and monetary budgets. So, I focused on other projects and became a spectator instead.

If you followed the event, you would have heard that no team completed the course. The Red Team from Carnegie Mellon University (CMU) got the furthest — 7.4 miles — when it high centered on a berm on the side of the road. They had a promising rig based on a highly modified HUMMER with plenty of computers and sensors. The closest competitor, SciAutonicsII, went only 6.7 miles. Most of the robots had a problem with a barbed wire fence that ran parallel to the road at the starting point of the race. That fence entangled several robots and fooled others. Though it was disappointing to hear that no one finished the race, it was encouraging that at least two had traveled over five miles without human

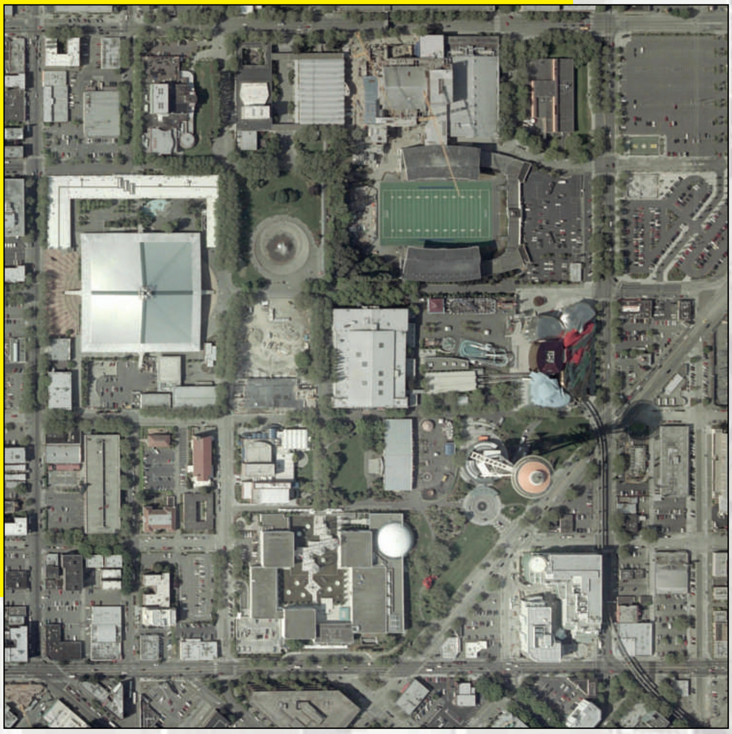


Figure 1. The Seattle Center, as seen from a satellite.

intervention. (*It sure is! - Editor Dan*)

On October 8, 2005, DoD plans to hold the event again with an additional $1 million prize and they plan to add another $1 million to the prize money each year, until someone completes the event.

Now, if only someone would broadcast it on live TV, then a lot of techno-geeks — like me — would be very happy. This is one of those contests that I wish I had both the time and money to compete in.

Recently, the Seattle Robotics Society (SRS, **www.seattlerobotics.org**) announced a new event called the "SRS/*SERVO Magazine* Robo-Magellan" contest. After last year's Robothon event (**www.robothon.org**), various SRS members decided that the Robothon needed a new robotics event that will challenge the advanced robot builders and stretch the average robot builder to try new things, while inspiring the beginning robot builders to move past simple line following and mini sumo contests. The various suggestions included a multi robot soccer using mini sumo robots, a balancing robot obstacle course, Robo-One style bi-ped boxing, and something that was strictly based on artificial intelligence (AI).

Then Doug Kelley — a board member of the SRS — took the bull by the horns and devised a set of rules for the Robo-Magellan event, which was loosely inspired by the DARPA Grand Challenge.

What makes this event really exciting is that — instead of sitting on the sidelines and watching the DARPA Grand Challenge — I now have an event I could build for and compete in.

Key Summary of Rules

The goal is simple: the robot must autonomously travel from point A to point B in the shortest time possible. The points will be physically marked with orange traffic cones and their locations will be given by GPS coordinates. They will be no further than 300 feet apart (as the bird flies), but there will be no direct line of sight between the start and finish cones.

An added level of complexity to the contest is that the robot must physically touch the orange cone at the final destination to stop the clock. There will be bonus way point cones along the way to help reduce the final score time — if they are touched — but they are optional and not required for completing the course.

Prior to the start of the event, the robot builder will be given the latitude and longitude coordinates of the starting, ending, and bonus way point cones. They will be allowed 30 minutes to walk through the course before the run; in this time, they can plan a route to the final cone (or to any bonus cones) and download this plan to the robot. Also, any hardware tweaks to the robot can be made. For example, GPS way points for the optimal path to complete the course could be logged and then downloaded into the robot.

There are only a few limitations regarding the type of robot that can enter. It must be fully autonomous. It must not weigh more than 50 lbs and cannot use a combustion engine — internal or external. It must have a fail safe mechanism to shut the motors off at a command from any judge at any time. The fail safe can be either a remote control or a physical control wire.

The robot must complete the course within 15 minutes and the robot will be given three attempts to complete the course. The lowest score will be counted.

One of the more important details I had seen was the "spirit of these rules" clauses. We robot builders are a creative lot. If you supply the problem, we will come up with the solutions to those problems — often in ways that go beyond the original idea that spawned the problem.

Sometimes, this is good; sometimes it just causes "unfair" short cuts to the event — like adding a 300-foot arm that reaches out and touches the cone at the end of the course. Wording is included that will allow the judges to keep the event on track as they meant it to be — an autonomous navigating robot contest. If you are unsure whether your interpretation of the design fits the rules, contact them in

Figure 2. Unexpected obstacles litter the area.

Figure 3. The expected obstacles are hard, too.

advance and they will help clarify the issue.

The most important detail is safety. The event will be conducted in an outdoor public area of the Seattle Center. The park-like area brings throngs of families and tourists and the SRS will not be able to restrict access to the grounds. The robot handler must be able to "control" the autonomous robot at all times so as to not endanger others. To take care of this, the rules require a "suitable fail safe stop mechanism" that must be fully demonstrated prior to starting the contest. If any spectator gets too close to the robot (or vise versa) while it is running the course, the operator must stop the robot.

Ideally, the robot should avoid these situations on its own. If the robot damages any property or poses any risk to spectators, its run is terminated.

Although not really a rule, a statement of fact is that you are liable for your robot and, "for any damage to person or property caused directly or indirectly." The event holders are not assuming responsibility for any entrants. The responsibility is ultimately up to the builder.

This article does not present all of the rules or areas covered therein; please contact **www.seattlerobotics.org** for the full set of rules.

Design Considerations

The course makeup is diverse and complex. The event is going to be held outdoors at the Seattle Center (Figure 1). For scale, look at the football field (top center). The distance between goal lines is 300 feet. The exact location among the 74 acre site will not been announced until 30 minutes before the start of the event. The complete campus is made of concrete, asphalt, and cobblestone paths around grass and pond areas that lead to the many venues. Some venues are outdoors — like the International Fountain and the Sculpture Garden — while many others are indoors (and, thus, are not options for the event).

The robot must deal with horizontal transitions from one surface to another (i.e., concrete to grass), but it has been stated that there will not be a requirement of going over any curbs (though your robot may choose to do so).

Even so, the ramps that are built into curbs can have an inch rise, so it would be best for the robot to at least be able to handle that. The many grass areas and shrubbery will require handling of the diverse traction issues that will be encountered.

There are also obstacles that the robot must contend with. The static ones include sculptures, curbs, buildings, benches, planters, and cloth items. Yes, cloth — but more on that later. The more challenging obstacles are the mobile and erratic types. These include automobiles, other robots, animals, and spectators. A park maintenance truck might not see the robot. A child might decide that the robot is a toy to get a ride on and chase after.

Now, this is an interesting case where the obstacle is actually chasing you! Most adults will probably avoid the robot as it makes it way around — especially if you put a warning light on it — but there is no guarantee of this. Methods will need to be built in to handle such erratic data and ignore it.

In my first trek through the campus, I was amazed at the number of personal items like backpacks, strollers, shoes, and even shirts put aside by the park goers (Figure 2). These would provide interesting obstacles to either avoid or traverse.

There are many things at the Seattle Center that make it an interesting park, but these same items turn it into a minefield for a robot. There are odd-shaped planters that not only have curved areas at the bottom, but also have cavities at about the height I planned to put all my proximity collision sensors (Figure 3). Then there are drop offs — like an 18 inch high curb, or stairs, or transitions from grass areas to the pavement.

On some walkways, there are rails that are made of wire and others of glass. Both have issues that must be addressed when building systems to acknowledge and avoid them.

You also have to contend with issues of light and sky

Figure 4. Both water and sand hazards await the competitors — better be prepared!

visibility. The varying foliage and covered areas will not only create dark and bright areas that may affect sensors, but they may even inhibit a good GPS lock if you rely on it. Lastly, you have the wonderful Seattle weather to contend with. There is always a good chance of rain in the fall, so the robot must be designed to operate in the rain. Well, Seattle rain is more like a drizzle, but don't ignore the possibility of a bright, sunny day that could saturate a sensor or two.

My Design Approach

I see the event as being solved with two primary behaviors or actions from the robot. These are navigating between cones, then targeting and touching them. While working on both of these primary behaviors, the robot will still need to do basic avoidance of objects — like those shirts and shoes I spoke of earlier.

Although GPS units are allowed on the robot, navigating the course can be done without GPS. The error inherent with GPS will often be higher than the width of many walkways. WAAS support will only give you less than 10 feet of accuracy if you can get the lock.

So, you probably can't rely on it if you do use one. Since they will be handing out the starting location and cone locations in lat-long coordinates, you could use these to provide distances and try dead reckoning methods. I hinted earlier about using a portable GPS unit to mark a path that could be downloaded into the robot and thus providing the path that could be followed, thereby avoiding the larger issue of which path the robot should take.

When the robot gets near the cones, it will need to actually touch the cone. This second primary behavior will need to use sensors to find the cone and then make its way to it. Since the cone will be a standard shape and a very unique color in the park (standard traffic cone orange), visual sensors seem to make a lot of sense. The rules state that the robot must actually touch the cone, so it makes sense to have sensors that actually sense touch — feelers or accelerometers — so you know that you actually did complete that goal.

Another design issue is how fast the robot should go and how long it must last on its power source. The maximum distance between the start and end cone will be 300 feet. This doesn't mean that the shortest path will be that long; most likely a much longer distance must be accounted for.

The speed of the robot is restricted to that of safe operational speed. Your fail safe switch must stay within range of the operator at all times. If it's a tethered design, this may limit the speed to as fast as you jog. (Hopefully, you don't think you can run after you bot at all times.) For wireless, you need to make sure it doesn't pass behind buildings and go out of range. You also don't want a robot that goes faster than its sensors can respond or it might run into something that could get you disqualified as being dangerous.

I consider a safe maximum speed to be around a brisk walk — three miles per hour or about four feet per second. If we assume the path is three times longer than the 300 foot maximum — about 900 feet — then we will minimally need to run on batteries for about four minutes. Since the robot must finish the course in 15 minutes, this gives me 11 minutes for pauses to allow for these mobile obstacles to move past my robot. To be on the safe side, though, you should plan on needing power for the full 15 minutes.

To handle the diverse conditions I have mentioned, I

Links

www.darpa.mil/grandchallenge

www.cmu.edu/cmnews/extra/031113_hummer.html

www.seattlecenter.com

www.robothon.com

www.terraserver.com

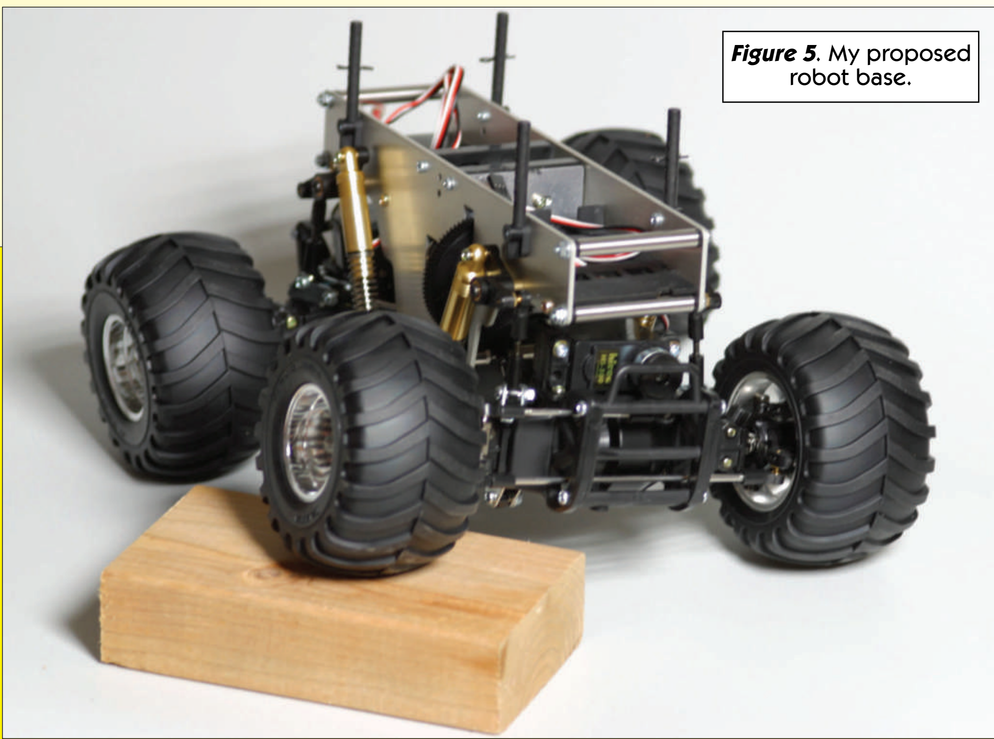


Figure 5. My proposed robot base.

wanted to have a base vehicle that could handle the various terrains and keep good traction. Although building a custom frame and running gear would be an interesting challenge, I wanted to focus on the core issues of this contest — navigation and detecting the orange cone. Like many of the DARPA Grand Challenge participants that chose to use something already designed and built, I also wanted to use something off-the-shelf. The weight restrictions made it an obvious choice to look at R/C (radio control) vehicles. I wanted something with a high ground clearance and good suspension articulation, as this would give me the ability for all the tires to remain in contact with the terrain — even on some of the roughest ground.

The local hobby stores had many candidates — 4 x 4 Monster truck kits that ranged from 1/10th scale to 1/18th scale. Most of them had four wheel drives and off-road high traction tires. Some of the more advanced kits used multiple motors for extra torque and speed. Tamiya had several interesting candidate robot bases to choose from. I ended up choosing the TLT-1 Rock Buster, since it was four wheel drive, along with a unique four wheel steering feature. The four wheel steering can help with turning in tight areas. Also, the price for this kit fit within my budget (Figure 5). Pictured is the vehicle traversing a 2 x 4 piece of wood; this vehicle can actually have one tire on the longer side and still keep all tires in contact with the surfaces — a very impressive range of articulation.

Next Time

This article was more of an introduction to the new "SRS/*SERVO Magazine* Robo-Magellen Contest," along with some of my personal observations of the Seattle Center and how they affect the overall design issues for my robot. I have shown my choice for a project base and, in next month's article, I will go into the sensor and navigation details of my robot entry for this contest. **SV**

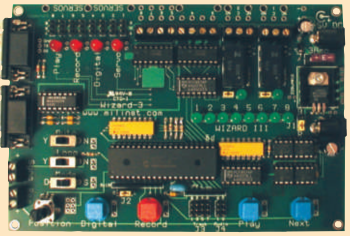
About the Author

Michael Miller has been doing software development for a diverse set of products for over 18 years — from clean room monitoring back in '80s, to operating systems through the '90s, to computer games of late. His latest work involved solutions that convert large scale geographic map data into an optimized data mesh for AI collision and path finding on a massively multiplayer role playing game and the systems that use it.

New Products

CONTROLLERS & PROCESSORS

Wizard-III Controller



Blue Point Engineering announces a new addition to its 20+ animatronic and robotic line of unique controller boards. The Wizard-III Controller is a unique stand-alone multifunctional board with many features. The Wizard-III Board records and replays 13 minutes of user-generated action for eight standard R/C type servos and eight digital outputs.

The board incorporates features such as looping playback action with variable delay between loop sequences, auto start-up on power up, quick connection interface terminals for sensors, power supply, and ability to daisy chain several controller boards together. Recording sessions for servos and digital channels are easily user programmed through onboard program keys and status LEDs to build a channel-by-channel basis of servo motion and digital output control.

No computer or software is needed and no complex programming is required. All previously recorded channels are replayed to aid synchronization. Playback of recorded programmed routines can easily be activated by onboard button, remote switch, or by various optional remote sensors.

Board Features:

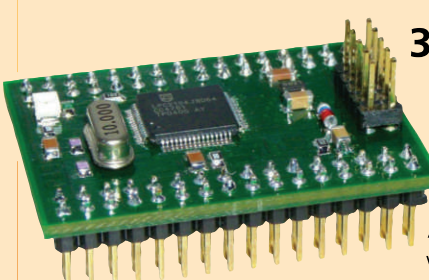
- Eight servo output channels, each capable of recording and replaying 13 minutes of recorded action.
- Eight digital switched ON and OFF channels capable of recording and replaying 13 minutes of recorded action with 4.5 volts DC at 100 mA output terminals.
- Onboard potentiometer to adjust servo positions during recording or to determine the time delay between replay loops during automatic loop play — adjustable between 5 and 65 seconds.
- Two digital channels configured with onboard selectable relays (relay rated 30 volt at 2 amp DC).
- Onboard NEXT, PLAY, DIGITAL, and RECORD user programming buttons.
- Record enable and disable jumper block to help safeguard programmed routines stored in EEPROM.
- AUTO-PLAY, LOOP-PLAY, and Servo Digital MODE selection switches for configuring control board functions and operation.
- Remote activation by switch or sensor with controller start-up options.
- Programming and board operation status Green and Red LEDs.
- The servo outputs provide standard pulse coded signals of between 1 msec and 2 msec duration, repeated every 24 msecs, making it suitable for all standard hobby 5 volt R/C type servos.
- EEprom containing the programmed data can easily be removed and copied for mass production or used with other Wizard controller products.
- Onboard support for other optional control boards (AC controller, motor bridge, solid state relays, sound boards, etc.).
- Comprehensive user manual with application examples.
- No computer or software needed to operate controller board.
- Easy onboard programming and edit features for user generated routines.
- Low cost, high quality, pre-assembled and tested board.
- Synchronization port to sound and other Wizard boards; multiple boards can be synchronized together, including puppet and other controllers.
- Board operates from a 5 to 12 volt DC power supply.
- Wizard Boards start at $85.00.

For further information, please contact:

Blue Point Engineering — Wizard Devices
213 Pikes Peak Pl.
Longmont, CO 80501
Tel: **303 • 651 • 3794**
Email: **bpe@bpesolutions.com**
Website: **www.bpesolutions.com**

Circle #30 on the Reader Service Card.

32-Bit!



Philips already offers them — powerful, small controllers with ARM7 core. Controllers with small dimensions are nothing new — but these are new. They are true 32-bit controllers. Their data buses are 32-bits wide, thus increasing processing speed. Their address

buses are 32-bits wide, thus eliminating the need for error-prone banking. Their timer registers are 32-bits wide, thus improving accuracy. PWM registers are 32-bits wide. These are things that embedded control programmers had to do without in the past. The 32-bit controller — LPC2194 — has the features listed below:

- 32-bit RISC architecture (ARM7TDMI-S)
- 256 Kbyte FLASH memory
- 16 Kbyte RAM
- Two UARTs
- Two SPI
- One IIC
- Four CAN controllers
- Four channel 10-bit A/D converter
- Two 32-bit timers
- Six 32-bit PWMs
- 46 I/O pins (including the above functions)
- Industrial temperature range
- LQFP64 package

For a quick start into the new 32-bit controller world Paul and Scherer offers development kits consisting of the C-compiler ECO-C-arm, documentation on CD, a board with the LPC2194 controller, an RS232 interface module, and the necessary cables. The price for the starter kit with ECO-C-arm demo compiler is EUR 112,00 (excluding VAT).

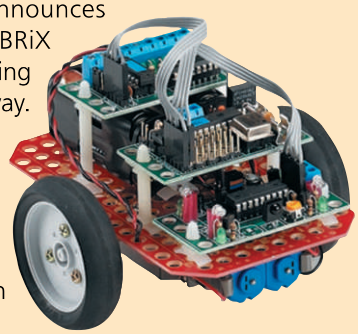
For further information, please contact:

MCT Paul and Scherer

Dorfstr. 4-6
17495 Ranzin
Germany
Fax: **+49 038355 68740**
Email: **sales@mct.net**
Website: **www.mct.net** or **www.mct.de**

Circle #78 on the Reader Service Card.

Build Robots a Whole New Way!



Mondo-tronics announces the new RoboBRiX Adventure Set for building robots a whole new way. Each RoboBRiX module contains a powerful PIC processor tuned for a specific task and each module thinks for itself, freeing you to work on the "big picture."

RoboBRiX communicate over standardized serial links — just plug them together! Their system of mechanical holes and spacers give extreme flexibility and they even accept LEGO® compatible components. Put the parts where you want and move them as needed.

The RoboBRiX Adventure Set includes the two-wheeled RoverBase, DualMotor1Amp motor driver module, IRProximity2 sensor module, PICBrain11 module with preprogrammed behaviors, plus battery pack for six AA cells (not included), cables, standoffs, and supporting parts. Configure the modules to build a wall follower, random bouncer, "attack dog," and more.

The set requires soldering and an Internet connection for online instructions. All parts are fully compatible with other RoboBRiX modules.

Available exclusively from **RobotStore.com** (item number 4-040), the RoboBRiX Adventure Set — priced at $89.95 — saves over 20 percent of the cost of buying the modules separately.

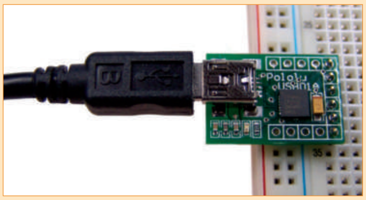
For further information, please contact:

Mondo-tronics Incorporated

124 Paul Dr. #12
San Rafael, CA 94903
Tel: **415•491•4600** or **800•374•5764**
Fax: **415•491•4696**
Email: **info@RobotStore.com**
Website: **www.RobotStore.com**

Circle #49 on the Reader Service Card.

USB-to-Serial Adapter



Pololu introduces their new USB-to-serial adapter for connecting microcontroller-based projects to personal computers. The diminutive unit measures under 1.0" x 0.7" including its connector, making it perfect for projects where space is a premium. For quick prototyping, the simple layout of the ground, transmit, and receive lines allows for easy mounting that takes up as little as four breadboard rows.

The USB adapter's drivers make it look like a standard serial port to the operating system. Therefore, the adapter can be used with existing software — such as servo controller interface programs — that are designed for traditional serial ports. Unlike most USB-to-serial adapters that require an additional RS-232-to-TTL converter, the Pololu USB adapter uses 3.3 V signal levels that can be connected directly to microcontrollers running at up to 5 V.

The adapter is compatible with USB 2.0 standards and allows baud rates of up to 921.6 kbps. Support is initially available for Windows 98 through XP; Mac and Linux support will follow shortly.

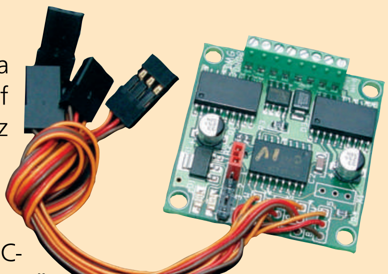
With the trend toward removing serial ports from new computers, the Pololu USB-to-serial adapter provides one of the most economical, small, and simple solutions to the common problem of interfacing small projects to PCs. The price for one unit is $23.00 with free shipping in the US.

For further information, please contact:

Pololu Corporation
600 S. Eastern Ave., Ste. 5-E
Las Vegas, NV 89119
Tel: **877 • 7 • POLOLU** or **702 • 262 • 6648**
Fax: **702 • 262 • 6894**
Email: **www@pololu.com**
Website: **www.pololu.com**

Circle #84 on the Reader Service Card.

Tiny Robot Motor Controller Packs Punch



SOZBOTS — a manufacturer of components for 16 oz robots — has just introduced the latest version of their small motor controller, SOZDSC-MX. The 1.5" x 1.5" controller can drive two motors for a robot — left and right — as well as a third motor from any hobby R/C radio. The SOZDSC-MX can run off 5-18 V battery power. The left and right motor drives are designed to drive your robot in typical tank steering and are rated for 5 amps peak; they are protected from over current, over temperature, and over/under voltage. Intelligent software can mix the two R/C channels so a single stick can drive the robot forward, in reverse, and in turns. The third channel is rated for 18 amps peak and is only meant to drive the third motor in a single direction. A fourth radio channel can be use for invert, in case your robot is invertible and flips upside down. To ensure precise control, the SOZDSC-MX can be calibrated with your radio system. The SOZDSC-MX weighs less than 1/2 oz.

For further information, please contact:

SOZBOTS
Email: **info@sozbots.com**
Website: **www.sozbots.com**

Circle #91 on the Reader Service Card.

ServoCenter 3.1: Module Allows Unprecedented Speed and Position Control

Yost Engineering, Inc., has introduced ServoCenter 3.1, an embedded R/C servo motor controller allowing independent control of both speed and positioning for up to 16 servos per board and 16 daisy-chained boards. Using only one serial port, unique speed and positioning parameters can be passed to each of 256 motors.

Unlike other controllers, this independent control of servo position and speed makes ServoCenter especially useful for applications such as robotics, animatronics, motion control, automation, retail displays, and other areas where independent, coordinated, fluid motion is desired.

A scaled positioning mode makes it easy to set maximum, minimum, and startup points. The speed control feature allows each servo to seek at a rate from 1% to 100% of its full speed.

Example programs are provided in GCC/Linux, QBASIC, VB.NET, C#.NET. VC.NET, VB 6.0, VC++ 6.0, and Turbo C, for both simple raw serial protocol and the included ActiveX control and DLL. An onboard regulator provides 6.0 V or 4.8 V with over-current, thermal protection, selectable baud rate, and flexible power options, including battery usage.

A complete package of ServoCenter, nine-pin serial cable, AC adapter, user's manual and programming guide, and software/examples CD is $69.95 (ServoCenter board only for $48.95).

For further information, please contact:

Yost Engineering, Inc.
630 Second St.
Portsmouth, OH 45662
Tel: **888 • 395 • 9029**
Email: **sales@YostEngineering.com**
Website:
www.YostEngineering.com/ServoCenter

Circle #102 on the Reader Service Card.

MECHANICS

HS-755HB Servo

Hitec's new sturdy HS-755HB is a giant-scale servo featuring exclusive "no wear" KARBONITE gear train technology. The HS-755HB is a great servo for demanding applications that require a large servo with more than 150 oz/in of torque.

The HS-755HB is offered with either the conventional "S" style connector (part #33755S) or the Futaba "J" connector (part #33755J) and uses the new KARBONITE composite gear train — a Hitec exclusive — that has eliminated lash and slop forever. KARBONITE gears are four times stronger than conventional white resin gears

and they are less likely to strip under the shock and loads that would usually break standard gears. These new gears were tested 250,000 times under load and showed no signs of wear. The result — greater reliability and control with Hitec's new KARBONITE gear train technology.

MAP	MSRP
$27.99	$45.95

Part Numbers
33755S ("S" connector)
33755J ("J" connector)

Size	Weight
2.30 x 1.14 x 1.96"	3.88 oz/110 g

Volts	Torque	Speed
4.8 V	152.75 oz/in	0.23 sec/60
6 V	183.31 oz/in	0.28 sec/60

For further information, please contact:

Hitec
12115 Paine St.
Poway, CA 92064
Tel: **858•748•6948** Fax: **858•748•1767**
Website: **www.hitecrcd.com**

Circle #99 on the Reader Service Card.

Revolutionary Instant Prototyping Material

ShapeLock introduces a revolutionary new material that enables experimenters to make super strong 3-D models and mock-ups fast. Rapidly create prototypes, parts, sculptures, brackets, housings, and molds with this new hand-moldable polymer. Stronger than most consumer plastics, this "Modeling Clay on Steroids" melts in hot water (160° F, 70° C) and then locks rigidly at room temperature. Shape and reshape it by hand; this material is re-useable and safe, as there are no chemicals or catalysts involved. This new product is immensely tough, lightweight, machine-able, paint-able, non-toxic, and inexpensive. The material is provided in an easy-to-use pelletized form and is available in a variety of pre-packaged experimenter sizes. Full instructions are included with every order, as well as access to extensive website-based information. Get your idea into shape today with ShapeLock.

For further information, please contact:

ShapeLock
557 E. Arbor Ave.
Sunnyvale, CA 94086
Tel: **408•551•9651**
Email: **Support@ShapeLock.com**
Website: **www.ShapeLock.com**

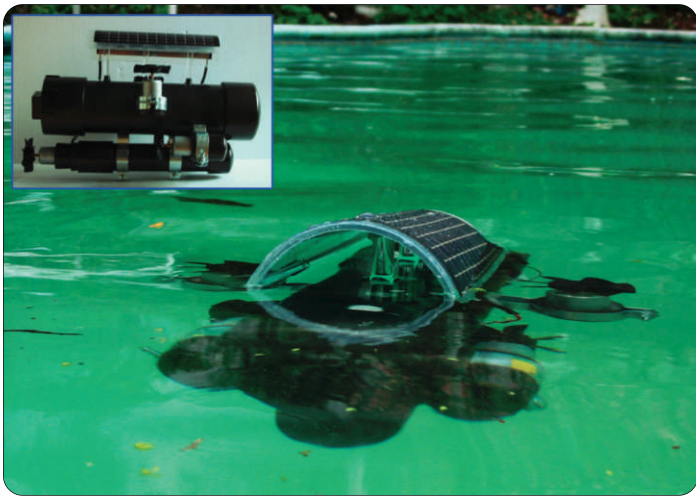
Circle #53 on the Reader Service Card.

Show Us What You've Got!

Is your product innovative, less expensive, more functional, or just plain cooler? If you have a new product that you would like us to run in our *New Products* section, please Email a short (300-500 words) description and a photo of your product to **Alexa@servomagazine.com**

MENAGERIE

Send us a high-res picture of your robot with a few descriptive sentences and we'll make you famous. Well, mostly. menagerie@servomagazine.com



Solar Powered AUV

Oliver Johnson, Salem, OR

I built this as a part of my high school research project on making cheap, small, long endurance AUVs. It is programmed to recharge itself through the solar panel and "knows" when to do this via voltage comparator chips. It submerges and navigates itself according to my program, which is on a Parallax BASIC Stamp 2.

I used lead shot to weigh the sub down so that it is neutrally buoyant (and close to sinking). Down-thrusting motors push the submersible below the water and — when it's turned off — it naturally bobs back up to the surface, exposing the solar panel for recharging.

The thrusters were made by customizing bilge pumps (used in boats) because they are already waterproof and the small ones (360 gallons per hour) are relatively cheap. The body of the submersible is made from PVC pipe. Marine sealant was used to make the sub watertight and protect the electronics.

The sub weighs about 17.5 pounds, displaces about 16.5 pounds of water, and has three forward thrusters, as well as two downward thrusters. The whole robot cost me about $150.00 in parts and materials.

Building Robots with Shape Memory Alloy

FIGURE 1. The Space Wings kit from Mondo-tronics includes shape-memory alloy wire and a specially designed actuator circuit.

The circuit causes the wires to slowly contract and expand, which makes the wings move.

by Gordon McComb

As early as 1938, scientists observed that certain metal alloys, once bent into odd shapes, returned to their original forms when heated. This property was considered little more than a laboratory curiosity at the time. The metal alloys were weak, difficult and expensive to manufacture, and broke apart after just a couple of heating and cooling cycles.

Research into metals with memory took off in the early 1960s, when William Beuhler and his team of researchers at the US Naval Ordnance Laboratory developed a titanium-nickel alloy that repeatedly displayed the memory effect. Beuhler and his cohorts developed the first commercially viable shape memory alloy or SMA. They called the stuff nitinol — a name derived from Nickel Titanium Naval Ordnance Laboratory. (Ordnance is the fancy sounding term for ammunition and other weapons used for warfare.)

Since its introduction, nitinol has been used in a number of commercial products. For example, several nitinol engines have been developed that operate with only hot and cold water. In operation, the metal contracts when exposed to hot water and relaxes when exposed to cold water. Combined with various assemblies of springs and cams, the contraction and relaxation (similar to that of a human muscle) causes the engine to move.

Other commercial applications of nitinol include pipe-fittings that automatically seal when cooled, large antenna arrays that can be bent (using hot water) into most any shape desired, sunglass frames that spring back to their original shape after being bent, and a novel anti-scald device that shuts off water flow in a shower — should the water temperature exceed a certain limit. Nitinol is also used in various medical devices, such as artery stints and even — ahem — implants for a certain portion of male anatomy. What will they think of next!?

Regular nitinol contracts and relaxes in heat (in air, water, or other liquid). That limits the effectiveness of the metal in many application where local heat can't be applied. Researches have attempted to heat the nitinol metal using electrical current in an effort to exactly control the contraction and relaxation, but, because of the molecular construction of nitinol, hot spots develop along the length of the metal, causing early fatigue and breakage.

In 1985, a Japanese company — Toki Corporation — unveiled a new type of shape memory alloy specially designed to be activated by electrical current. Toki's unique SMA material — trade named BioMetal — offers all of the versatility of the original nitinol, with the added benefit of near instant electrical actuation. BioMetal and materials like it — Muscle Wire from Mondo-Tronics or Flexinol from Dynalloy — have many uses in robotics, including novel locomotive actuation. From here on out, we'll refer to this family of materials generically as shape memory alloy or simply SMA.

Basics of SMA

At its most basic level, SMA is a strand of nickel titanium alloy wire. Though the material may be very thin (a typical thickness is 0.15 mm —

slightly wider than a strand of human hair), it is exceptionally strong. In fact, the tensile strength of SMA rivals that of stainless steel: the breaking point of the slender wire is a whopping six pounds. Even under this much weight, SMA stretches little. In addition to its strength, SMA also shares the corrosion resistance of stainless steel.

Shape memory alloys change their internal crystal structure when exposed to certain higher-than-normal temperatures. This includes the induced temperatures caused by passing an electrical current through the wire. The structure changes again when the alloy is allowed to cool. More specifically — during manufacture — the SMA wire is heated to a very high temperature that embosses or "memorizes" a certain crystal structure. The wire is then cooled and stretched to its practical limits. When the wire is reheated, it contracts because it is returning to the memorized state.

Although most SMA strands are straight, it can also be manufactured in spring form, usually as an expansion spring. In its normal state, the spring exerts minimum tension, but — when current is applied — the spring stiffens, exerting greater tension. Used in this fashion, SMAs become an "active spring" that can adjust itself to a particular load, pressure, or weight.

Shape memory alloys have an electrical resistance of about 1 Ω per inch. That's more than ordinary hook-up wire, so SMAs will heat up more rapidly when an electrical current is passed through it. The more current, the hotter the wire becomes and the more contracted the strand will be.

Under normal conditions, a two to three inch length of SMA is actuated with a current of about 450 milliamps. That creates an internally generated temperature of about 100-130° C; 90° C is required to achieve the shape memory change. Most SMAs can be manufactured to change shape at almost any temperature, but 90° C is fairly typical for off-the-shelf material.

Excessive current should be avoided. The reason: Extra current causes the wire to overheat, which can greatly degrade its shape memory characteristics. For best results, current should be as low as possible to achieve the contraction desired and no more. The excess current is dissipated as heat, and the higher heat will more rapidly degrade the functionality of the wire. Shape memory alloys will contract by two to four percent of their length, depending on the amount of current applied. Maximum contraction of typical SMA material is eight percent, but that requires heavy current that can — over a period of just a few seconds — damage the wire.

Using SMA

Shape memory alloys need little support paraphernalia. Besides the wire itself, you need some type of terminating system, a bias force, and an actuating circuit.

Terminating

Terminators attach the ends of the SMA wires to the support structure or mechanism you are moving. Because SMAs expand as they contract, using glue or other adhesives will not secure the wire to the mechanism. Ordinary soldering is not recommended, as the extreme heat of the soldering can permanently damage the wire. Many of the SMA experimenter's kits come with pre-terminated wire. These are handy when you're just starting out with shape memory alloy. You can start playing moments after you take the wire out of the package.

For self-terminating, the best approach is to use a crimp-on terminator. These and other crimp terminators are available from companies that sell shape memory alloy wire (either in the experimenter's kit or purchased separately). Ring terminals — designed to attach an electrical wire to a screw terminal — are ideal for anchoring an SMA wire. Crimp the SMA wire into the terminal, then secure the terminal using a small (2-56 or 4-40) screw.

You can make your own crimp-on connectors using 18 gauge or smaller solderless crimp connectors (the smaller, the better). Although these connectors are rather large for the thin 0.15 mm SMA, you can achieve a fairly secure termination by carefully folding the wire in the connector and pressing firmly with a suitable crimp tool. Be sure to completely flatten the connector. If necessary, place the connector in a vice to flatten it all the way.

Bias Force

Apply current to the ends of an SMA wire and it just contracts in air. To be useful, the wire must be attached to one end of the moving mechanism and biased at the other end. Besides offering physical support, the bias offers the counter-acting force that returns the SMA wire to its limber condition once current is removed from the strand. Without the bias, the SMA wire may simply sag. Useful bias mechanisms include a small spring (metal or rubber) or a weighted object.

Actuation

SMAs can be actuated with a 1.5 volt AAA penlight battery. Because the circuit through the SMA wire is almost a dead short, the battery delivers almost its maximum current capacity. The average 1.5 volt alkaline penlight battery has a maximum current output of only a few hundred milliamps, so the current is limited through the wire. You can connect a simple on/off switch in line with the battery.

The problem with this setup is that it is wasteful of battery power and, if the power switch is left on for too long, it can lead to some damage of the SMA strand. A more sophisticated approach uses a pulsing circuit — such as a 555 timer IC — that automatically shuts off the current after a short period of time. Such circuits are the fodder of any SMA demonstrator kit or book, so there's no need to duplicate them here.

Even more sophisticated drive circuits are used to achieve specialized activations. For these, a PIC or other microcontroller can be used to produce complex timings, with dampened rise and fall times. The microcontroller is connected to the SMA wire through a fairly simple transistor or buffered gate output in order to provide adequate drive current.

The benefit of using a microcontroller is that the drive timing can be easily changed simply by rewriting the software. You can also more easily accommodate sensory feedback. For instance, you might connect an electronic thermometer to the microcontroller in order to sense ambient temperature. Assuming that the SMA wire you are driving is air cooled, you can compensate for the speed of wire relaxation by sensing the ambient temperature around the wire.

Shape Memory Alloy Mechanisms

With the SMA properly terminated and actuated, it's up to you and your own imagination to think of ways to use it in your robots. One typical application of using an SMA wire is in a pulley configuration. Apply current to the wire and the pulley turns, giving you rotational motion. A large diameter pulley will turn very little when the SMA tenses up, but a small diameter one will turn an appreciable distance.

You can also attach a length of SMA wire in a lever arrangement. The metal strand is attached to one end of a bell crank. On the opposite end is a bias spring. Applying current to the wire causes the bell crank to move. The spot where you attach the drive arm dictates the amount of movement obtained when the SMA contracts.

SMA wire is tiny stuff and you will find that the miniature hardware designed for model R/C airplanes is most useful in constructing mechanisms. Any well-stocked hobby store will carry a full variety of bell cranks, levers, pulleys, wheels, gears, springs, and other odds and ends to make your work with SMA more enjoyable.

Sources for Shape Memory Alloy

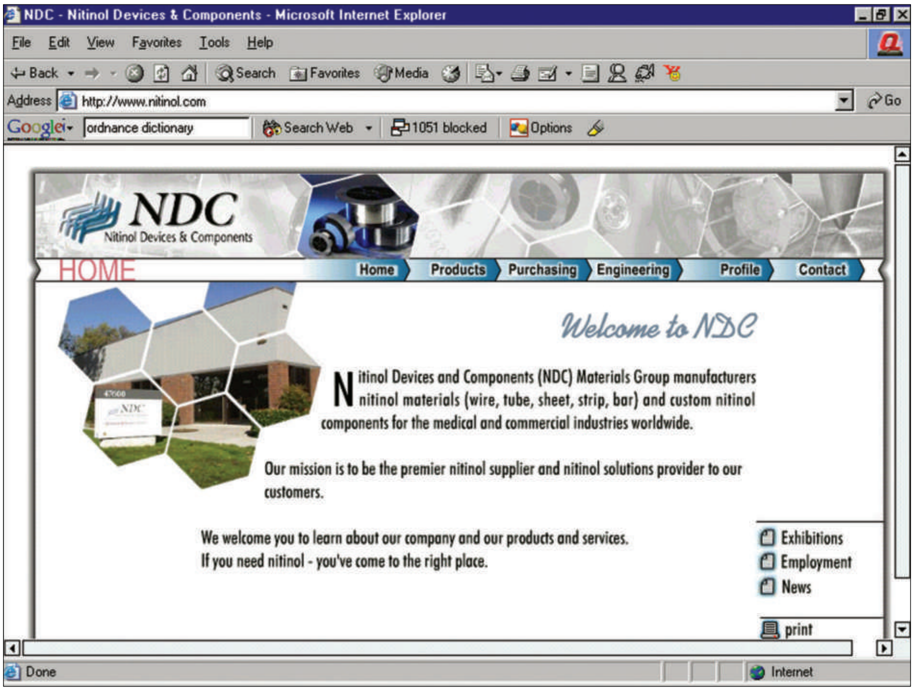
Dynalloy, Inc.
www.dynalloy.com

Dynalloy, Inc., is a manufacturer of shape memory alloys specially made to be used as actuators. They offer wire by the meter, sample kits, and pre-crimped Flexinol (for easier attaching to things).

Images SI, Inc.
www.imagesco.com

Online retailer of various robotics

FIGURE 2. Nitinol Devices & Components is one of several specialty manufacturers of nitionol wire, tubing, and other products.



parts, including nitinol kits and components.

Memry Corp.
www.memry.com

From the website: "Memry Corporation is a recognized leader in the development, manufacturing, and marketing of semi-finished materials (wire, strip, and tubing), components and assemblies utilizing the properties exhibited by shape memory alloys — in particular nickel titanium (nitinol or NiTi)."

Mondo-tronics, Inc.
www.musclewires.com

Major seller of shape memory alloy materials, as well as books and kits. Sold through distributors or the company's RobotStore.com web outlet.

Nanomuscle, Inc.
www.nanomuscle.com

Nanomuscle is a specially-manufactured shape memory alloy that does the job of a miniature solenoid. Apply voltage and the Nanomuscle actuator contracts several millimeters; remove voltage and the device relaxes.

A developer's kit is available and the company provides onsite purchasing, but only in quantities of 25 or more units.

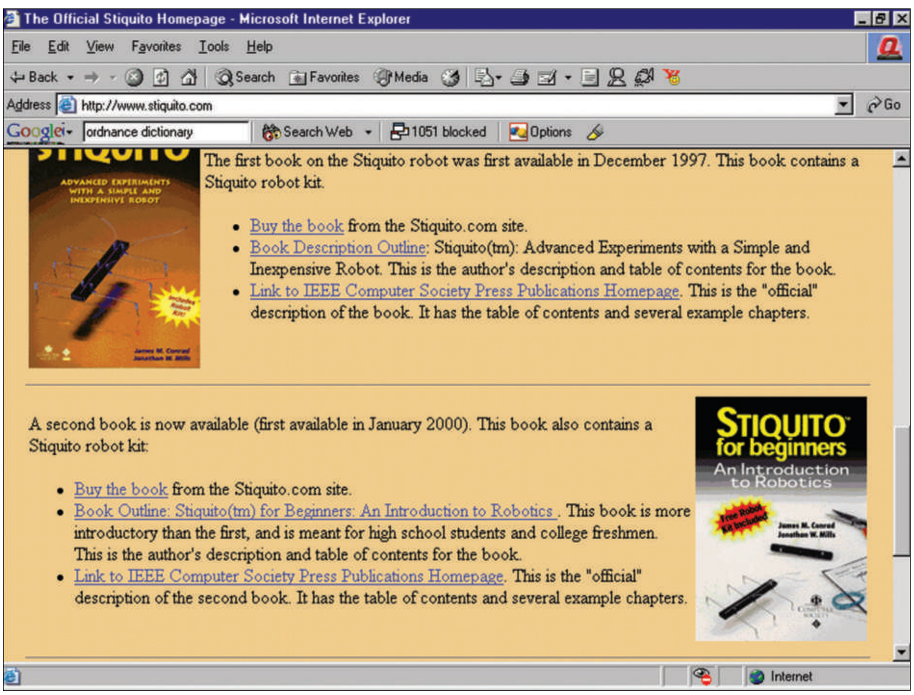
Nitinol Devices & Components
www.nitinol.com

In the words of the web page: "NDC is a leading supplier of nitinol materials (wire, tube, sheet, strip, and bar) and components to the medical and commercial industries worldwide." Datasheets of products are provided for download.

Shape memory and Superelastic Technologies
www.smst.org

Volunteer organization of industry professionals dedicated to disseminating technical education of shape memory and super elastic properties, especially nitinol alloys. Conference procedures and links to companies and other organizations involved in shape memory alloys.

FIGURE 3. Stiquito.com provides sales and support for a series of books on constructing a small, six-legged robot using shape memory alloy wire.



Shape memory Applications, Inc.
www.sma-inc.com

Manufacturer of shape memory alloy materials, including tube, sheet, and foil applications.

Special Metals Corporation
www.specialmetals.com

Makers of shape memory alloy materials. Technical documents available for download in Adobe Acrobat PDF format.

Stiquito
www.stiquito.com

Stiquito is a small and simple robot that uses shape memory alloy (SMA) wire for movement. This is the official Stiquito page — maintained by author Jim Conrad — and supports the product and several books written about it.

Toki Corp.
www.toki.co.jp/BioMetal/index.html

From the English translation of the web site: "BioMetal is one of Ti-Ni based shape memory alloys; however, its properties are specially arranged for use in our own manufactured actuators. The material being metal, it provides smooth and life-like (biological) movements; thus, it has been named 'BioMetal.' BioMetal is offered in the form of a thin wire (BioMetal Fiber), which, in facilitating electrical current passage, performs best in tensile-directional usage." Much of the website is in Japanese, with additional technical information available in that language. SV

About the Author

Gordon McComb is the author of the best-selling *Robot Builder's Bonanza*, *Robot Builder's Sourcebook*, and *Constructing Robot Bases*, all from Tab/McGraw-Hill. In addition to writing books, he operates a small manufacturing company dedicated to low cost amateur robotics, **www.budgetrobotics.com** He can be reached at robots@robotoid.com

How to Be Perfect

Even if Perfection Is Over-rated and Otherwise Impractical

by Allan Comeau, Ph.D.

At the center of Steven Spielberg's film, *AI, Artificial Intelligence*, which follows Stanley Kubrick's initial concept and the story he developed from Brian Aldiss's *Super-toys Last All Summer Long*, is David, a 60-pound, 11-year-old boy who is ready to love and be loved by his new parents — Monica and Henry Swinton. As the movie tagline reveals: "His love is real. But he is not."

David is unique; he is a different kind of human creation. He is a robot, programmed to be capable of loving his adoptive parents who are admonished not to speak the seven-word love activation code, "unless you mean it."

As circumstance — or bad fortune — would have it, speak it they did (well, to be precise, only the mother does), and David's love becomes activated tragically, for, in the end, he "loses" his beloved parents when his biological half-brother returns to the family. By film's end, he is, in effect, orphaned, waiting in a state of robotic hibernation until he is awakened many years later to yet another world.

David, like Pinocchio — a running theme in the story — is a man-made creation who longs to become a real boy. The theme is not only evident, it is an explicit component of the film. As in Pinocchio's tale, it is not enough that his puppet-maker Geppetto both fashioned and loved him: some special intervention is also needed.

In both "Pinocchio" and *AI*, we find the device of the Blue Fairy — a personification of the feminine, mother-like spirit. Pinocchio's courageous love earns him his humanity. For David Swintin — while his undying architecture sustains his undying love — alas, this love is no longer returned by his programmed love-object "mother," except through the grace and compassion of his eventual salvagers.

When I first considered this film's plot development, I didn't give much thought to the penultimate sequence — when David is forever separated, relationally orphaned, from his mother and father — but, when I viewed it from the perspective of perfection — although from a human point of view — it occurred to me to pause and reflect on the *AI* implications of the concept of perfection. For what are robots, if not componentially "perfected" humanoids?

The First Robot

I hope that at least one reader will be informed and know that the word robot was coined by Czech writer Karel Capek and first appeared in his 1923 play, *R.U.R., Rossum's Universal Robots*. In *R.U.R.*, robots are artificial persons — soulless, mechanical servants. Suffering from radical curiosity, I dug into my reference books to find that robot is derived from the Czech robota, connoting forced labor and servitude.

Readers familiar with German will know the related word Arbeiten, meaning work. Oddly, this word's root, arbi, is etymologically kin to orb, an Indo-European root, which is also the source for the word orphan. I sure would like to know whether director Spielberg or screenwriters Ian Watson and Brian Aldiss were cognizant of that when they left David in suspension for all those centuries.

Perfection

Approaching the concept of perfection, I am tempted to adopt a layman's sense of dichotomous extremes, positing on one side that which is perfect and, on the other, all else. In cognitive behavioral therapy, perfectionism — along with all-or-nothing thinking and jumping to conclusions — is considered a cognitive distortion.

Along with dysfunctional beliefs and inadequate coping skills, cognitive distortions lead us down the path to anxiety, depression, and other unpleasant states. (Readers may wish to review my essay on cognitive therapy at my website, **www.DrComeau.com**). What prompted me to write about perfectionism in the first place was to investigate what parts of perfectionism we might be better off without, retaining any components that are still useful.

The Perfect Handshake

In robotics, as in aerospace technology, one of the standard problems encountered in

the design of grasping or holding mechanisms is the need to be able to exert sufficient force to hold, but not damage, the target object. The problem is difficult enough when the object is solid — even if it has "handles.". It is exponentially more challenging when the target object is breakable and even worse if it happens to be living, with both soft and semi-hardened edge features.

How would robotic arms and hands determine when a squeeze is a hug, as opposed to a crushing vice-grip? Tackling this question, even philosophically, requires an acknowledgment of the accountable uncertainties. Even humans can hurt each other with a handshake or an unexpected or unwelcome hug, unaware of either our strength or the expectations of our recipients.

In human contact terms, it is essential for both (of two) parties to negotiate (or suffer) such intimacies as hugs or other forms of pressing the flesh. Considerate people try to both offer and respond to each other's physical presence and comfort preferences by continuously reading each other's tactile and other cues. Long standing couples or friends can establish an "affection algorithm," which provides a set of preexisting conditions for the "perfect," appropriate hug, handshake, or other form of intimacy.

Perfection and Procrastination

How often do we get stuck in the beginning, middle, or even near the end of a task and fail to go any farther because some part of ourselves takes control and won't let us do our best and finish what we started? One of the primary causes of procrastination — putting off till tomorrow what we might very well be able to do today — is the fear that our efforts will result in failure or, at best, incomplete success. For some of us, the idea that we might make a single mistake, exposing both our external and internal flaws, can stop us in our tracks, resulting in the realization of our own worst fears of failure.

I can imagine that — as soon as the first mousetrap was conceived, built, and brought to market — some enterprising soul then conceived "a better mousetrap," only to be surpassed by "the perfect mousetrap," and so on. Somehow, the idea of perfection must itself be a secondary conception, following the logic of: "mousetrap first, perfected mousetrap to follow."

In Western philosophy, going back to Plato, things, in the form of tangibles, were considered but failed approximations of "the ideal," perfect forms, which themselves could only be conceived of in thought or meditation. As good a mousetrap as could be made in Plato's time would never have been called perfect, as perfection was not a quality dared to be shared by any inhabitants of the material world. Notwithstanding these and other technical obstacles, let us now reach for the stars and dare seek our destinies in humanly realizable perfection.

The Roots of Perfection

Looking at the word perfect, we find the components per, meaning "thoroughly," and fect, from the Latin facere, meaning "to make or do." So, at face value, being perfect means just doing whatever you do thoroughly. Unfortunately, there are some added meanings to perfect, including being faultless or flawless — and here's where it gets difficult to be perfect.

The Perfect Medical Student

One could say that, if it weren't for flawlessness, it would be a lot easier to be perfect. I remember treating a young medical student some years ago. He was getting good grades, but he was not getting As. One might think that a B-plus in the microbiology or gross anatomy exam would be well received by nearly any student, but not for Jim, as I'll call him.

"Whatever I don't know might come back to haunt me," he worried. I asked how this might be so, though I had started to get a sense of where he was going with this. "Treating patients who are ill is serious business," he continued. "If I make one critical mistake, someone could lose his or her life and whose fault would it be, but mine?"

I tried to talk him down from what could eventually become — if it was not already — a crippling, obsessive perfectionism. I told him about the dentist I had learned about from my own dentist. That doctor was so obsessed with getting each piece of work right that he would sometimes redo a filling or an enamel repair over and over again, crippling his office staff and backing up his schedule for hours.

I warned him that, if he took his desire to be a good and effective doctor too seriously and too compulsively, the inevitable outcome would be that he'd have to limit his schedule to seeing one patient per day — and he'd better have a "perfect" nurse by his side to catch all his mistakes.

We all know that too much attention to both important and unimportant details can result in impaired concentration. In this potentially excellent doctor's case, his efforts to learn everything, instead of focusing on the important things, resulted in his appearing not to know enough and in his morale and self-confidence bottoming out.

It may seem a technical point, but I think that people can be a lot more perfect than they typically imagine themselves to be. If we could put that flawlessness on standby and focus on the task at hand, most people could finish a lot more things, doing them thoroughly and carefully, and, as a result, they'd be a lot more "perfect."

It doesn't seem that hard to do whatever you do thoroughly and completing what you start — especially if that's the fast track to perfection. Just don't stop until the job is done and don't turn your work in until you've double-checked it against your own (or perhaps your bosses') standards. Remember that practice makes perfect and, with practice, almost anyone can be perfect.

What about those people who are perfectionistic to a fault, like Dr. Jim, who can't tolerate a single mistake or error?

One thing that we have to realize is that we are all what I call "perfectly

imperfect." Our brains and nervous systems are designed to perceive and react to only a small — but useful — part of reality. What we can see, we call the light spectrum; what we can hear is the sound spectrum, and so on. The brain, like a wondrous machine, does what it is designed to do and more, all a consequence of evolution, across the millennia of our species.

So, why don't we make the most of our sensory and mental capacities and interact with the world and each other, just doing the best we can? Someone once called to my attention a Balinesian saying, "We have no art; we do everything the best we can." It is this sort of sentiment that is, I think, the antidote to crippling perfectionism.

Adding the "ism" to perfect doesn't help things very much. The airplane pilot or the ship's captain makes many course corrections before he or she can land safely at the given destination. With this understanding, the perfect achievement can be seen as the result of many adjustments, made possible by many sightings and soundings. It is not flawless as much as it is flexible and guided by some vision or plan.

Our bodies are designed with large motor and small motor capacities. To lift a cup, I must use large muscles to bring my hand to the lifting field, then I can employ smaller, finer muscles to grip, pour, or take a sip. What a job it would be to have to use only large or small muscles to complete the whole task!

Things Can Be Perfect, but People Don't Have to Be!

One of the problems that comes with trying to be perfect is that even the idea of perfection — implying flawlessness — requires that we have a standard and then measure ourselves or something according to that standard. There's a logical or categorical flaw at work here: While things can be perfect because they can conform to a humanly specified set of standards, people cannot and need not be perfect (to a perfect standard) because people — by nature — are constantly changing, hopefully evolving, and certainly unpredictable most — if not all — of the time.

Here are some ideas for moving beyond perfectionism to the perfect realm of thorough doing and being:

- Continue to learn and improve.
- Do whatever you do according to your own standards — and, if you don't have standards, choose some.
- Make the extra effort to complete what you have started.
- Consider all experiences of failure to be a lesson and welcome the opportunity to do better next time.
- Realize that the quest for perfection has more to do with you and your own development than it does with the task that you are trying to do perfectly at the moment.
- Focus more on the doing and the exercise itself in self-improvement and less on the task or outcome.
- Remember that, in sports like baseball, the all-time home run hitters are often also the all-time strikeout leaders.
- Keep your sights on long-term growth and forget about the imperfection of a single failure.
- Don't define yourself by your failures. Even better, don't define yourself by your successes.
- Do define yourself by your willingness to keep your word and thoroughly do what you've promised to do.

Going back to my original thesis about "Pinocchio" and *AI*, both Pinocchio's and David's tasks reflect our own: to become real, we must develop our own potential. In a sense, this is how we must perfect ourselves. Our tasks cannot be completed in isolation because essential components can only be accessed through interactions with others — sometimes with teachers — but most often with fellow travelers, each of us possessed of an increasing sense of our own emerging humanity. In the final analysis, the best approach is do the best you can. **SV**

GEERHEAD

by David Geer
geercom@alltel.net

Danger vs. Mean Burrito, aflame!
Photo by Eric Stoliker.

Megabyte vs. ShinSplitter
Photo by Andrew Rossol.

Mammoths and Dinkies 2

"Pick on someone your own size!" say these Super Heavyweight Combat Bots! The 16 oz Antweights say likewise for Mammoths and Dinkies 2 — more of the biggest and smallest robots you've ever seen!

Introducing, in This Corner ... !

These fighting robots run up to 340 lbs, but no more. The technical specs for qualifying fighters are pretty much the same from class to class. Your bot gets thrown into whatever class it makes weight for on arrival.

The famed *Battlebots* show and its resulting phenomenon started off the Super Heavyweight class around its second season, when the event was filmed in Las Vegas, NV.

This photo from the Tensilica Pro-Am shows Andy Sauro's Danger vs. Ross Hironaka's Mean Burrito. Photo by Eric Stoliker.

Here We Go!

Even big bots get all torn up over competition, especially by Megabyte, whose byte is worse than his bark. Megabyte is the work of three builders who make up the Robotic Death Company.

Carl Lewis, John Neilson, and John Mladenik — owners of Robotic Death Company — and Megabyte have had a stroke of beginner's luck — or just good craftsmanship and hard competition — as they've won competitions in their first year and are Heavyweight and Super Heavyweight champs. That's a first.

What Qualifies as a Super Heavyweight?

The Super Heavyweight class is actually around 340 lbs, but Megabyte is only 220 lbs. So, what gives? Well, Megabyte fights in the Heavyweight class, but is also allowed to fight in the Super Heavyweight competitions. How

does Megabyte do this?

Megabyte is the current National Heavyweight Champ and has won first place in both of the tournaments it fought in as a Super Heavyweight. At Mechwars 7, Megabyte defeated the Super Heavyweight Champ Merr Mad. At the Southwest Division Championship at the R/C Expo in Anaheim, CA, it went five to zip against a robot that was over 50% heavier.

The Super Heavyweight class — also referred to as The Big Boys — has a greater challenge in store when "Megabyte's Evil Twin" is completed. This new bot, created especially for the Super Heavyweights, will be the same size as Megabyte, but will weigh in at a whopping 320 lbs!

Soon, the Robotic Death Company will be showing up to fight in both the Heavyweight and Super Heavyweight classes in the same competitions.

What Makes Megabyte So Deadly?

The Robotic Death Company's star bot spins a heavy outer shell equipped with tool steel blades. The blades "byte" into opponents at up to 200 mph. The shell itself is titanium. Needless to say, Megabyte not only wins matches, but often comes out looking much better than its prey. It pays to be tough-skinned.

Megabyte wins via the fighting robot version of a technical or full knock out, leaving few matches with decisions. In a knockout, the defeated bot is pretty much disabled, but there is a tap out option, where the bot resigns before taking any more of a beating. Many of Megabyte's opponents have been benched for life.

Few bots make for as many keen stories as big fighting robots.

"I Remember the Time We Fought Good Ol' Abe, Uh, LBE!"

Unbeknownst to me and maybe you, too, competitions exist where you can fight "multi-bot." What's that mean? I'm about to tell you.

While at the Southwest Division Championship, Megabyte owners John Neilson and John Mladenik pit their mega-spin monster against Little Blue Engine (LBE) with the aid of two 12 lb robots — named Romulus and Remus. The two small bots rode in on Megabyte's back, willing to dare LBE along with Megabyte.

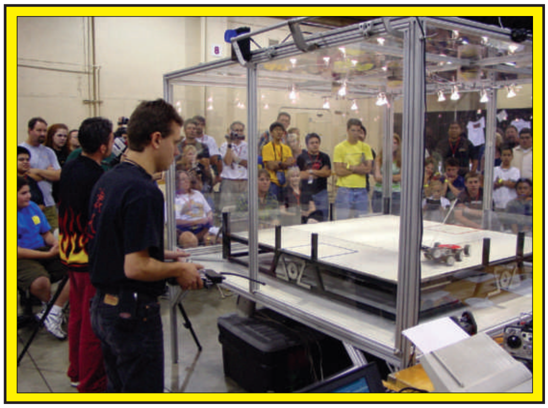


Photo 1. Peter Abrahamson drives Tsunami in the Sozbot arena. Photo courtesy of Sozbots.

Mladenik had built the smaller robots for his two young daughters, who had already faced competition using them. Now, these robots were entering the big ring, triple teaming with Megabyte.

The two little bots are wedge-style creations, used in this match to get under Little Blue Engine to slow it down. About midway through the battle, John Mladenik drove Romulus into Little Blue Engine to wedge under it, but LBE pushed the robot back into Megabyte, destroying the little bot. Romulus was actually thrown by Megabyte's spinning shell and teeth into a wall across the combat area.

First Timers — at One Time or Another, Weren't We All?

A precursor to Megabyte, Rambite was a 60-pound spinner that first competed in 2001 — the same year that Robotic Death Company was founded. That year, Rambite won two fights against Nsynerator.

Motorama? What a Kill'a!

Robotic Death Company's Killabyte (30-pounder) fought a 24 lb wedge (get this — named Janet Reno's Dance Party) at Motorama 2004. Janet Reno's Dance Party was kind of rough as it took a pounding before striking back. Then the wedge rammed Killabyte's own forks up into the blade, which stopped the bot from spinning.

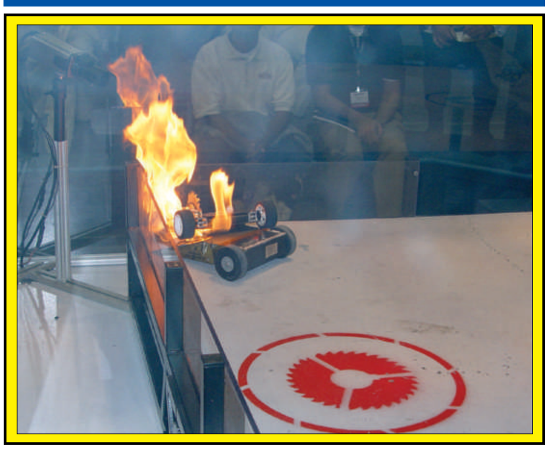
Killabyte was shoved around until it became stuck in the side of the arena. Janet Reno's Dance Party rammed it to set it free. Janet Reno's Dance Party was going to win in a huge upset when, suddenly, she stopped working. (Anyone smell a political parallel here?). Anyway…

Killabyte won by knockout, though John feels his opponent deserved the victory. Killabyte took second overall in that tourney.

What could make that little old Antweight think he can kick butt up on a stage? Let's see.

Well, at Sozbots events these are the rules: You have to be 16 oz. You can't blow things up, shoot free flying objects, spread chemicals or liquids, or use netting or other entanglement devices. However … you *can* use flamethrowers.

Photo 2. Mean Burrito burns the bottom of Tsunami. Photo courtesy of Sozbots.



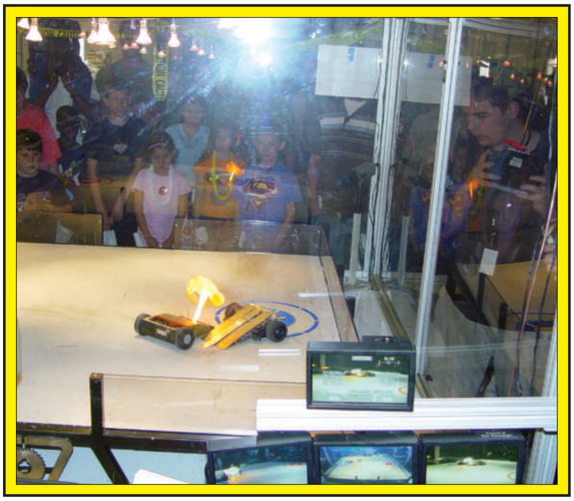


Photo 3. Mean Burrito attacks Pnu Jimmy. Photo courtesy of Sozbots.

The Birth of Sozbots

The Sozbots company was formed by Patrick Campbell, Eric Stoliker, Brian Roe and Peter Abrahamson as a branding effort within the Antweight class of robots. The four roboticists got the idea at one of the national fighting robot events, just as Antweights were coming to fruition.

Watching some young people play with the Antweights there looked incredibly fun (and — what a relief — easy, compared to the weight and cost of building and fighting the big boys).

They decided to sponsor their own events. The small bots fight with the same weapons as any others, though keeping under the 16 oz weight requirement is the real challenge.

Builder's Briefing ...

Another Antweight aficionado — Jim Snook — gave us some more dish on Antweights. Antweights have to fit in a 12 inch diameter circle. In addition to flamethrowers, they have come to do battle equipped with lifters, flippers, saws, and pinchers.

Jim's own Antweights are Jim's Bot and Chigger. Like many Antweights, these are modified R/C toys. Jim's Bot uses a large titanium plate in front as a pushing device. Chigger is armored and is also a pushing robot.

Fundamental to converting the toys into the real fighters is making them lighter. (There's that 1 lb limit challenge again. Can you believe the toys are actually heavier than the real thing?) You also have to decrease noise; the motors can be loud — not quite like the Jetcar races mind you — but loud enough.

You also have to change the wiring to use a different voltage. While making them lighter, you often have to make them larger, as they are generally undersized.

Getting Started

Like many others, Jim Snook started out with a toy and rebuilt it. (He went down in weight, up in size, made changes inside and out, and then showed up to compete.) Knowing how to do that and getting a good regular R/C controller — or better yet the PlayStation robot controller from Sozbots — is all you need.

Jim works on his bot at the dinner table at home. Just make sure you build according to rules and specs for competition and find out where they're competing next in your area. At the very least, competitions can be found in several cities in states near you.

You can always start your own meets and competitions, too. Competition rings are only four by four steel sheets with Lexan around them.

What Gets Roboticists Excited About Antweights?

Ross Hironaka's first competition event was a year and a few months

Sound Bytes on Megabyte, Part 1 — Losses (Awww!)

Megabyte vs. Bambulance

Here's the champion. (Pronounced "champ-ene" or "champ" by announcers in the old days ... boxing, of course.) Megabyte holds 29 wins and 4 losses — a 20 to 3 record as a Heavy Weight and 9 to 1 as a Super Heavyweight.

According to coach John Mladenik, losses were often due to Megabyte fighting too well — as if you can be penalized for that! Well, it appears you can be. Yes, in one match, Megabyte tore a hole in the arena; well, that was considered a dangerous hazard, so Megabyte was handicapped the rest of the match.

Being forced to reduce his punch (spinning speed) by 25%, the champ was bounced off the wall by a rubber Rambo named Bambulance in the first 20 seconds. Officials called the fight for safety — not Megabyte's, mind you, but that of the crowd. What a shame to lose a decision for being too powerful.

Megabyte faces the Sewer Snake

There's no shame in losing a fight where everyone is at their level best — right, roboticists? One match where Megabyte was defeated was with Sewer Snake. The Sewer Snake was relentless, taking its own hits and giving up bites of itself to its front end and a wheel. No offense to Megabyte, but it kind of reminds you of Mike Tyson, except that, here, those bites are legal!

Broken, but still moving, Sewer Snake won because, well, near the end Megabyte wasn't — still moving, that is. Megabyte melted a Kevlar belt taking all the belts and blows from that double S — the Sewer Snake.

Megabyte and the Shin Splitter

With the previous model of titanium shell on Megabyte, the Shin Splitter was able to use its large, tooled steel blades to cave in the shell and stop it from spinning. A little more caving in from the Shin Splitter and the drive wheels were, well, kaput!

ago in Santa Barbara, CA.

The fighting robot competitions on TV led Ross to go live and get the rush of a real competition. Finding heavy bots to be too big of an investment, Ross went at first to watch in February, 2003, at Steel Conflict in Pomona, CA. There, Ross discovered the Antweights. They were in his range for ease of build, use, transport, and cost.

The clincher was that flamethrowers were permitted on Antweights! From this germination came Mean Burrito — Ross' flame throwing robot.

Mean Burrito Goes Down in Flames and Ross Has Fun Doing It

The first time out with Mean Burrito, Ross learned a valuable lesson. We must test our robots well in advance of our competitions. Ross had tested Mean Burrito's component mechanisms, but hadn't tested it assembled as a finished robot. He did, however, make sure it got a trial — just before he was about to leave for the competition!

Whoosh!

Well, the flamethrower worked alright — sort of. The robot ignited and erupted into a raging inferno — instant meltdown. Ross patched up his pal with glue and tape and headed into battle anyway. No guts, no glory. Fortunately, Ross had managed to keep the robot's guts inside it long enough to fight — and have a great time doing it — against competitor Andy Sauro.

Photo 4. VDD launches his opponent over the wall. Photo courtesy of Sozbots.

In an awesome combat venue, the crowd was screaming as Sauro's robot, Danger, pummeled Mean Burrito. There were big screen TVs broadcasting the action to the crowd; people were hanging on every move, every blow.

The crowd started shouting, "Flame him, flame him!" Suddenly Mean Burrito's pilot lit up and all the force of that great initial blast of fire shot out from the flamethrower and Sauro's bot retreated. Mean Burrito caught fire internally. No one stopped the fight as Mean Burrito haplessly

Sound Bytes on Megabyte, Part 2 — Wins (Ohhhhh!)

Megabyte has zero losses since correcting the last of the flaws that held him back — that titanium shell. Zero losses, but one, that is, as a Super Heavyweight at the Mechwars 7.

In top shape since September 2003, the champ went on at that same Mechwars event to KO all other comers in the Super Heavyweights and, finally, to defeat that same bot that had defeated him earlier — Merr Mad — in the finals, winning the Mechwars 7 Super Heavyweight title.

Photo 5. VVDD sends Shenanigans flying.
Photo courtesy of Sozbots.

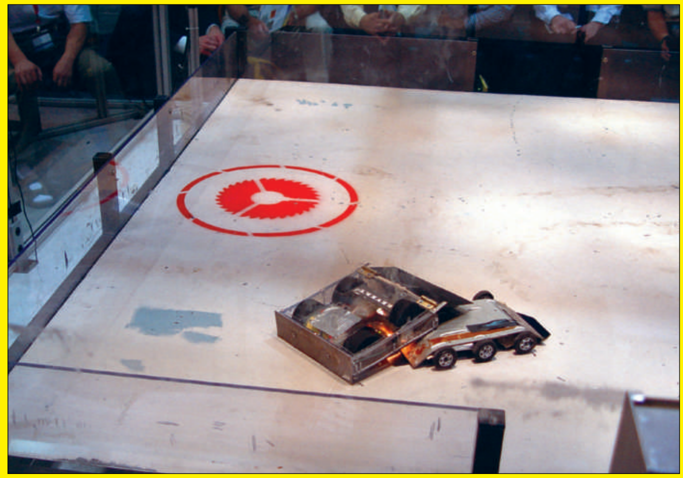


Photo 6. Incinerator lights Revert on fire.
Photo courtesy of Sozbots.

moved out of the ring to eventually be extinguished.

Mean Burrito — the Ingredients

Mean Burrito is made of drives from hacked toy robot motors. Originally, Mean Burrito was constructed of the cheapest materials in order to save on what would initially be a learning experience in Antweight robotics. Ross knew the machine would need to be rebuilt and repaired again and again.

Mean Burrito has been through several versions, last competing as version 3.2. Ross has used the controller from Sozbots and graduated through several types of batteries.

What Will Be the Flame?

Is it a cheap trick? Only butane is allowed for flamethrowers. Flame throwing devices must be unmodified. Upon learning the ins and outs of "unmodified," Ross put two butane cigarette lighters in parallel — valves wide open. The setup gives a colorful effect for the crowd.

Taking the Bows

The roboticists I talked with for this column agree that this weight class is a lot of fun, while remaining fairly inexpensive. There are no age limits, so young people can participate and there is a hoard of weapons available to build into these bots. It's all the fun of a Motorama or Steel Conflict without all the hassle incumbent with huge robots. **SV**

Resources

This site includes video of Megabyte, Romulus, and Remus facing off against Little Blue Engine.
www.saidin.com/robot/SC5_results.htm

Where to catch the Super Heavyweights in action. See the builder's database for upcoming events at:
www.buildersdb.com

Check out the Robot Fighting League at:
www.botleague.com

Browse the online forums at
http://forums.delphiforums.com/CJRC
and
http://forums.delphiforums.com/THERFL

Here are some shots of Megabyte putting holes in the steel edges of an arena.
www.roboticdeathcompany.com/john/TSN2003
(pictures 38 through 46).

Catch video of the Starhawk fight here:
www.chaosengineering.com/SDC

Coming events include this one in Minnesota (call the NPC Charity Event or see this link for details).
www.nelsenmachine.com/MMER/NPC%20Charity%20Open.htm

More videos can be found at
www.chaosengineering.com/SDC and
www.saidin.com/robot/SC5_results.htm

We're not done yet. Here are some Antweight videos:
http://homepage.mac.com/roninsfx/Menu8.html

And don't forget to stop by
www.steelconflict.com and
www.sozbots.com for
more on these Mammoths & Dinkies, the Super Heavyweights, and the 1 lb wonders called the Antweights!

More Antweight images!
www.robolympics.net/photos/ant01.jpg

http://team-corrosive.i8.com/cgi-bin/i/ROBOTS/10.jpg

http://team-corrosive.i8.com/cgi-bin/i/ROBOTS/13.jpg

http://team-corrosive.i8.com/cgi-bin/i/ROBOTS/Gilroy/108_0809.JPG

EXPERIMENTING WITH SHAPE MEMORY ALLOYS

Build a NanoMuscle Actuator Switch

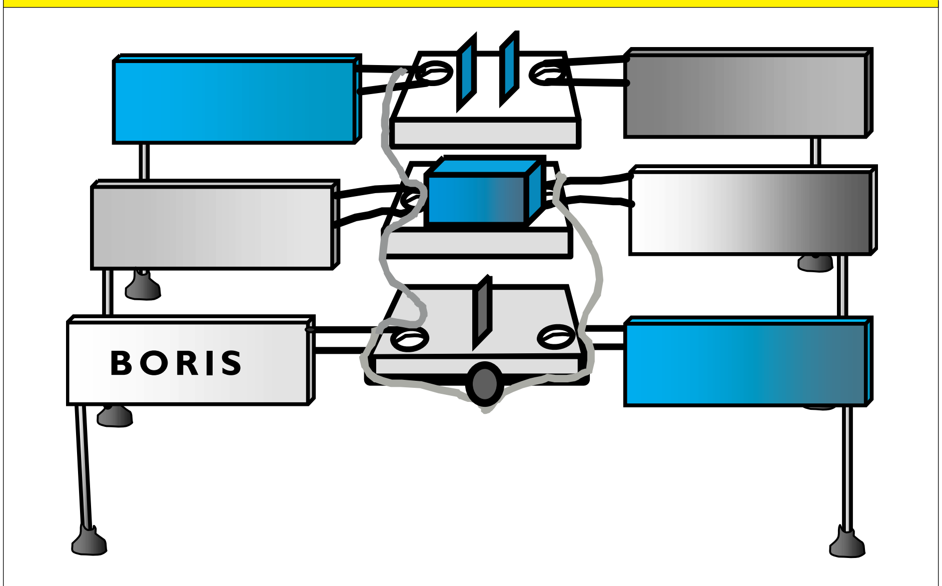
Part 1 — by Don Wilcher

Making small mechanical actuators using Nickel Titanium Alloy (Nitinol) wires has been a fascination since its discovery at the Naval Ordinance Laboratory (now called the Naval Surface Weapon Center). The Naval Surface Weapon Center found that the Nitinol could be stretched from its original shape when heated. An electric current from a small, 1.5 V battery could accomplish this deformation quite easily; thus, small mechanical actuators could be fashioned using this unique material.

Since the discovery of this material, there have been several books written describing how small mechanical actuators and robots can be built using Nitinol wire. Two books that come to mind are *Stiquito™ for Beginners: An Introduction to Robotics* and *Stiquito: Advanced Experiments with a Simple and Inexpensive Robot*. Both books are written by James M. Conrad and Jonathan W. Mills and are published by the IEEE (Institute of Electrical and Electronics Engineers) Computer Society.

Two famous "micro" robotics kits that evolved from the use of Nitinol wire were Stiquito and Boris. These robots were constructed using small plastic boards, aluminum tubing, 28 AWG and 34 AWG wires, music and bus wire, and Nitinol wire. Once the material was constructed, the finished product was a small, insect-like robot whose movement could be controlled using a 9 V battery.

Figure 1. Boris, courtesy of the Richfiles.



Boris (Figure 1) was a more complex micro robot compared to its sleek cousin, Stiquito (Figure 2), because of the sectional joints that made up its body and legs. In constructing the Stiquito robot (see Figure 3), I found it to be quite challenging because each leg had to be formed correctly along with routing the Nitinol wires through the holes of the plastic board for attachment to movable members using cut aluminum tubing sections and music wire.

For some electronics hobbyists, mechanical building may not be their forte and the thought of building micro parts for electric actuators is a total turn off. Do not despair; there is another alternative solution: NanoMuscle Actuators (NMA). The information in this article will explain how NMA can be used as a small micro electromechanical system (MEMs) to make electronically-controlled mini actuators for mechatronic-based projects and experiments. Before the hands-on discussion of NMA can take place, let's explore the physics behind Nitinol, which is the core material behind the NMA.

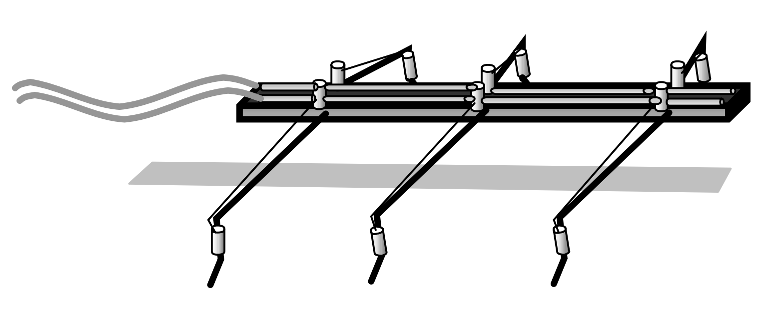


Figure 2. Stiquito, courtesy of the Official Stiquito Home Page.

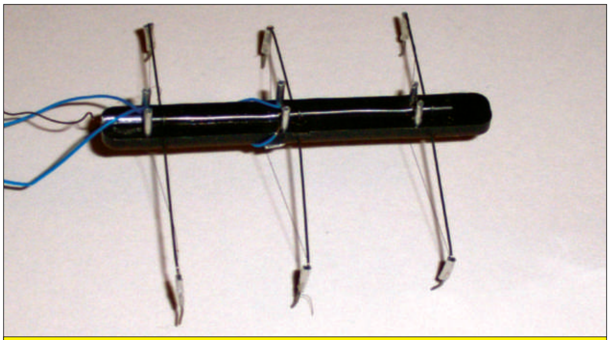


Figure 3. Stiquito, built by the author.

Nitinol Basics

As mentioned in the introduction, Nitinol wire is a shape memory alloy wire made from nickel and titanium. Shape memory alloy (SMA) is a "smart" material that can change shape or state with the application of stimuli such as heat or electric current [1]. Nitinol's small diameter wire contracts like a muscle when electric current flows through it. The heat dissipated ($P=I^2R$) assists in this morphic state change. The ability to flex or deform is a physical attribute of SMA. SMAs can dynamically alter their internal structure at certain temperatures. A bias or counterforce is needed to return the SMA or Nitinol wire to its original length or shape.

Mechanical motion conversion is accomplished by heat dissipation — a product of the electric current and the Nitinol wire's resistance. The heat that allows a light bulb to glow is based on temperature elevation. Instead of illumination being produced, as it is with an incandescent light bulb, the Nitinol wire contracts by several percent of its length when heated and stretches out as it cools down. The Nitinol's movement is silent, smooth, and strong; it occurs through a solid state phase of the SMA's restructuring.

When nickel and titanium atoms are available in the alloy, the material forms a crystal structure, called a *lattice*. The SMA structure is capable of changing from one lattice orientation to another. This transformation process moves the crystal between two forms – austenite and martensite — if heat is added or removed. An appropriate temperature level is required in order for the martensitic transformation to take place within the SMA's structure. If the crystal form's transformation temperature is higher than the martensite, then the SMA is in

What Is Mechatronics?

Mechatronics is a field that integrates traditional mechanical, electrical, and computer engineering and focuses on the synergism between actuators, sensors, controls, computer architecture, software, and knowledge[2]. Mechatronics is not new; basically, it is an interdisciplinary field that integrates the latest techniques in precision mechanical engineering, controls theory, computer science, and electronics to the design process to create more functional and adaptable products[3]. The word "mechatronics" was first coined about 30 years ago by an engineer employed at Japan's Yaskawi Electric Company to describe computer controls in electronic motor applications.

The quarterly *Transactions,* published jointly by the IEEE/ASME in March 1996, covers the interdisciplinary field of mechatronics. *Transactions* covers a range of related technical areas, including modeling and design, system integration, actuators and sensors, intelligent controls, robotics, manufacturing, motion control, vibration and noise control, micro devices, optoelectronic systems, and automotive systems.

A mechatronics system consists of a closed loop system of components working in a dynamic mode of operation. Basically, the mechatronic systems equation is based on:

system = mechanics + electronics + software [4]

To expand on this equation, a mechatronics system can consist of the following sub components:

- sensors
- signal conditioning and amplification
- analog to digital converter
- computer hardware
- control software
- digital to analog converter
- actuators

Depending on the complexity of the system, either several or all of the subcomponents can be employed for the complete mechatronics design.

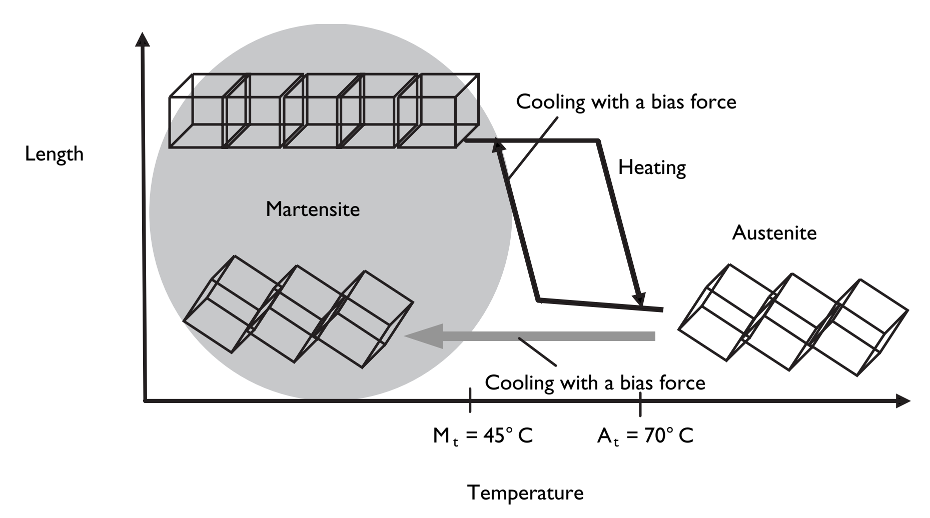
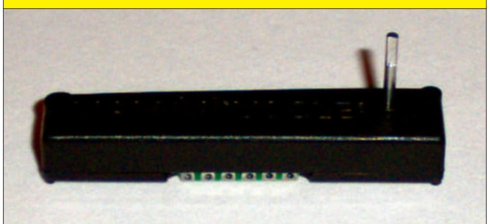


Figure 4. The transformation graph for Nitinol wire.

an austenitic state.

Figure 4 shows Nitinol's transformation states. Once the SMA is in this transformation state, the material exhibits high strength and doesn't deform easily. It is this heating and cooling process that allows the mechanical motion of the Nitinol wire to exist, thus making the material applicable to small electric actuators. With the basic understanding of the Nitinol wire physics known, a micro electromechanical system (MEMs) actuator can be developed.

Figure 5B. An actual NanoMuscle rotary actuator.



What Is an NMA?

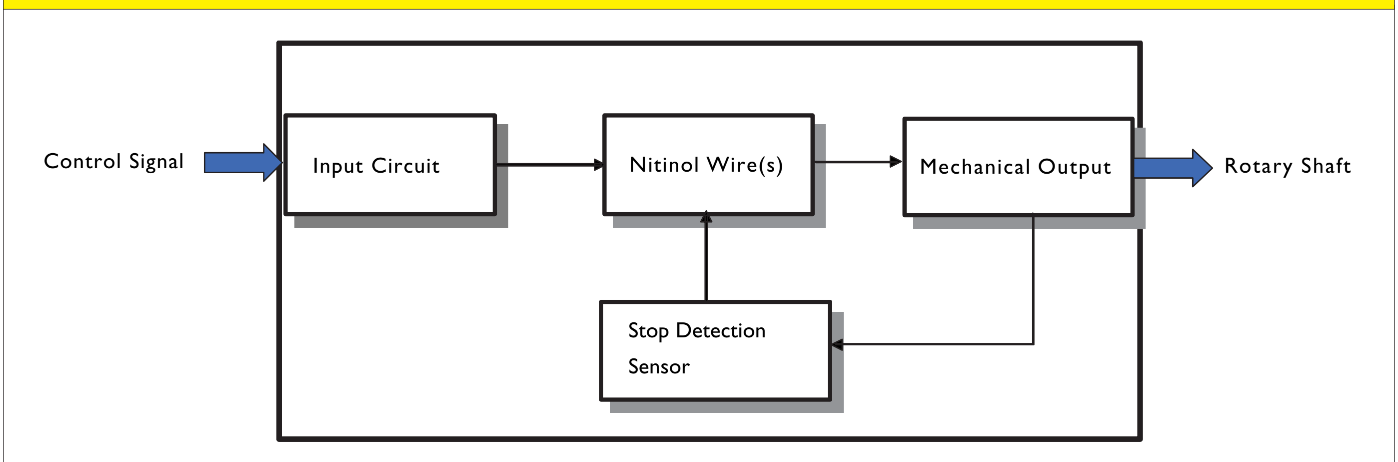
The core material of the NanoMuscle Actuator is the Nitinol or shape memory alloy wire. When an electric current passes through the wire, the SMA contracts because of the resistive heating of the conducting material. With the electric current removed, the SMA returns to its original length. The application of a tensile force assists in returning the SMA to its original state. By controlling the electric current of the SMA wire, the speed of contraction can be adjusted to a final position.

Basically, the SMA wire doesn't need to return to its original length, but can be adjusted to any intermediate length for mechanical motion control.

A key element of the NMA is an embedded Digital Interface (DI). This embedded circuit interface consists of a layer of electronics and sensors packaged inside of the MEMs device. The DI provides control and status feedback in a form that can be wired directly to a microcontroller or microprocessor without any additional electronic circuits or electrical devices.

Figure 5A shows the system block diagram of an NMA and Figure 5B shows an actual NanoMuscle rotary actuator. This mechatronics approach to system integration produces a lower cost and smaller package solution that

Figure 5A. System block diagram of the NanoMuscle actuator.



can be added to a digital system in less time than an electromagnetic equivalent — like a motor or solenoid. The key features of the NanoMuscle's DI consist of:

• An automatic adjustment of power levels to produce the requested movement.

• An integrated stop detection circuit that can signal the controlling microprocessor or microcontroller when the NMA is fully extended or contracted.

• A high density Flex Circuit connector is available to fit various product applications.

The NMA with the integrated DI is advantageous to a standard electric motor because a gearbox, rotary to linear converter, end stop sensor, and H-bridge assembly is required for electromechanical actuators and motion control applications. The NMA is a self-contained MEMs package that allows it to be used in a fraction of the time and at a substantially lower cost than traditional, electromagnetic, motor-based systems.

So, how easy is it to control an NMA? A microprocessor or microcontroller can be programmed to control an NMA directly. Digital and analog circuits can be designed to control the NMA, as well. The following paragraphs will explain how a simple DC network circuit can be used as an electronic control circuit for switching an NMA device.

An Electronic Circuit for NMA Control

A simple DC network circuit can be used as an electronic controller for switching NMA devices. In designing electronic circuits, input and output requirements must be available to capture the specific feature or function of the intended product.

The NanoMuscle Company that makes the MEMs-based actuators and motors have several products to use in commercial and toy products.

This project will use the NanoMuscle rotary actuator RS-125-CE device. Figure 7 shows the pinout for a NanoMuscle rotary actuator RS-125-CE device. Data sheets are a great resource for designing electronic circuits. The documents contain key electrical parameters that will allow the electronic designer to create circuits for specific product applications.

The key electrical requirement for the NMA electronic controller is the holding current. Holding current refers to the amount of electricity needed to keep the electrical load energized after it is switched on. The holding current requirement specified in the data sheet for the NanoMuscle rotary actuator RS-125-CE is 75 mA. Therefore, the DC network circuit must be capable of providing 75 mA of constant switching current to operate the RS-125-CE device. Figure 6 shows a schematic circuit for the electronic controller.

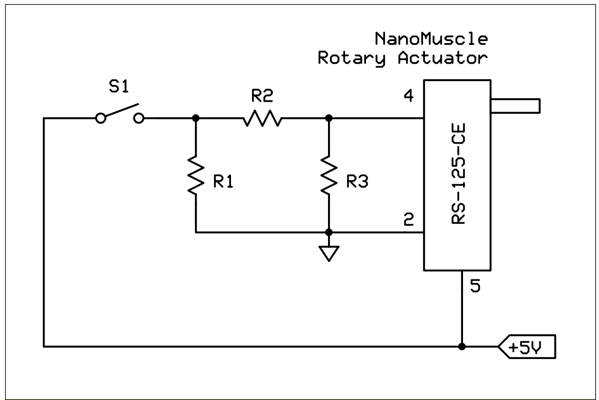
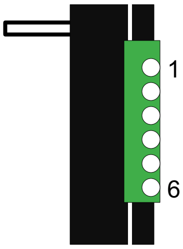


Figure 6. The NMA electronic controller schematic.

To help in the product development of the NanoMuscle rotary actuator remote control, Ohm's Law can be used in designing the DC network circuit for the electronic controller.

In designing circuits, certain assumptions are made about the target design. To determine the value of resistor R1 in Figure 6, the electrical current assumption for the electric switch contacts is 1 mA of the wetting current.

Wetting current is used to prevent



Pin Number	Function	Description
1	PO	High output when fully extended/unwound
2	GND	Ground
3	P100	High output when fully contracted/wound
4	Control	Command input — Setting this pin high will cause actuation.
5	Vsma	Power
6	Position	Position feedback pin (output)

Figure 7. Pinout for the NanoMuscle rotary actuator RS-125-CE.

oxidation (rust deposits) from accumulating on the electric switch contacts. When the switch contacts are closed, a small amount of current will flow through them, thus preventing oxidation buildup. This 1 mA wetting current requirement is used to calculate the value of R1 using the Ohm's Law equation:

$$R1 = V_{R1} / I_{R1}$$
$$R1 = 5\ V / 1\ mA$$
$$R1 = 5K$$

To calculate R2, the voltage drop across the resistor value must be known.

The target control voltage to turn on the NMA device is 4.5 VDC. Therefore, the voltage drop across the resistor is determined by:

$$V_{R2} = V_{R1} - V_{R3}$$
$$V_{R2} = 5\ V - 4.5\ V$$
$$V_{R2} = 0.5\ V$$

To calculate the R2 resistor value, Ohm's Law is used. The value of R2 equals:

$$R2 = V_{R2} / I_{R2}$$
$$R2 = 0.5\ V / 75\ mA$$
$$R2 = 6.67\ \Omega$$

Finally, R3 will be determined using Ohm's Law, as well. The value of R3 equals:

$$R3 = V_{R3} / I_{R3}$$
$$R3 = 4.5\ V / 75\ mA$$
$$R3 = 60\ \Omega$$

Armed with the calculated resistor values, a prototype circuit can be breadboarded. After building the circuit on the breadboard, testing it simply requires a 5 VDC power supply to energize the controller.

By pressing and holding the electric switch in the closed position, the NMA rotary actuator's shaft should rotate in a 60° angle and stop. Releasing the switch will allow the shaft to return to its normal or home position.

If this actuator is not moving with the command control signal from the electric switch, turn off the power supply and check for wiring mistakes, as well as incorrect resistor values. After the correction(s) have been made, retest the electronic controller again.

Congratulations, you have just stepped into the world of nano technology! Next month, we will use the electronic controller and the NMA device to build a MEMs-based actuator switch to control a small DC motor. **SV**

Resources

Here is a list of resources where the electronics hobbyists can find additional information on the material presented in this article.

[1]J. Ogando, "Intelligent Fasteners," *Design News Magazine*, October 20, 2003
www.designnews.com

[2]Macchrone, C. "Mechatronics: Breaking the Boundaries of Traditional Engineering," *Solid Works*
www.mechatronics.me.vt.edu

[3]Ashley, S. "Getting a Hold on Mechatronics," *Mechanical Engineering Magazine*, ASME, 1997

[4]Asulander, D.M., Kempf, C.J, *Mechatronics: Mechanical System Interfacing*, Prentice Hall, 1996

Boris micro robot homepage
http://richfiles.solarbotics.net/BORIS.html

Stiquito micro robot homepage
www.stiquito.com

NanoMuscle motors and actuators homepage
www.nanomuscle.com

About the Author

Donald Wilcher is an adjunct professor, technical writer, and electrical engineer with 18 years of automotive electrical/electronics experience. He has written two books: *LEGO Mindstorms Interfacing* and *LEGO Mindstorms Mechatronics*, both published by McGraw-Hill. His papers and engineering education activities can be found on his website **www.familyscience.net**

Adding Some Analog to the JStik

by D. Jay Newman

We live in an analog world, yet our processors are digital. Most microcontrollers have built-in Analog to Digital Converters (ADCs). However, sometimes these built-in ADCs are not enough because they don't have enough resolution, the speed needed, or enough channels available. Sometimes you choose — like I did — a microcontroller without an ADC at all.

This article will describe a SimmStick I/O board designed to be used along with the JStik (or, indeed, any SimmStick processor board). One of the major shortcomings of the Ajile processors used in the JStik and JStamp is that they do not include any Analog to Digital Converters (ADCs). This project seeks to remedy this by using three Microchip MCP3208s to provide a total of 24 channels of 12-bit ADC.

I'll admit that my reason for wanting 24 ADC channels is that I'm using this card for robotics. However, this same technique could be used anywhere you need as few as four channels of ADC (with the MCP3204 chip).

The Hardware Choices

Because the processor I'm using (a Systronix JStik) comes on a JSimm (SimmStick) board, I built my I/O board for the SimmStick. The JSimm bus has defined lines for SPI and various general I/O pins.

I choose to use the Microchip MCP3208 eight-channel, 12-bit ADC chip for several reasons:

1. It has eight channels per chip.
2. The power use is low.
3. It communicates with the host via SPI (Serial Peripheral Interface), which I already knew how to use.
4. It does the conversion fast.
5. The chip can either use eight single-ended channels or four differential channels.
6. It's inexpensive.

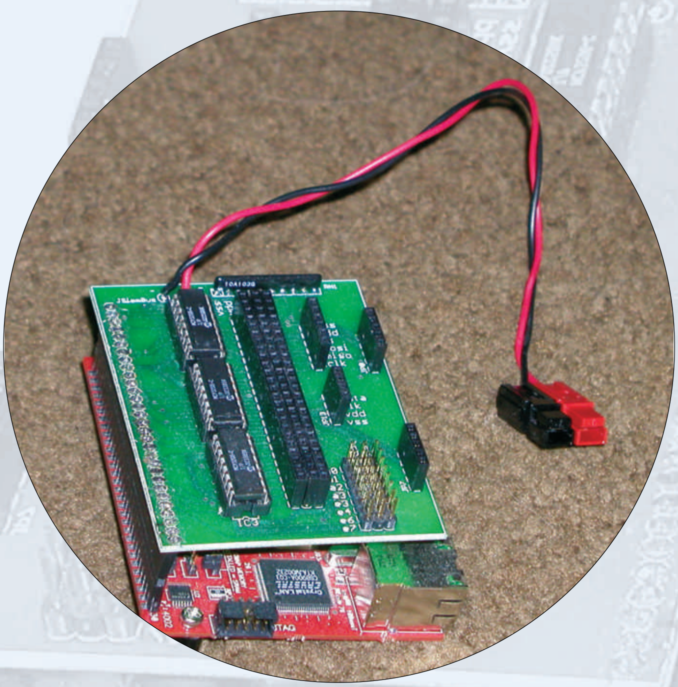
While there are many other chips available, the combination of the above factors had me using the MCP3208. In fact, the main factors for me were easy availability, ease of use, and SPI communication.

Because the JStik is the brains of my operation, I was pretty much forced to use the JSimm bus. On the other hand, the JSimm bus has all that I need to control everything I want.

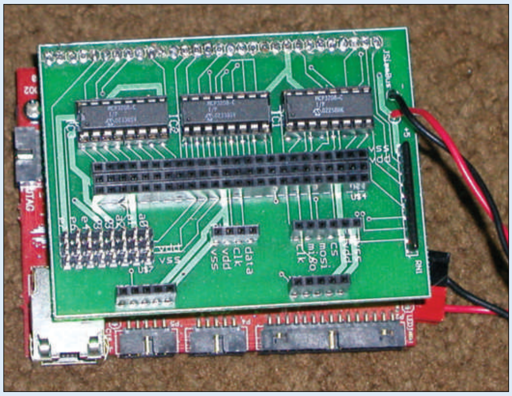
The Board

My first version of this circuit was done with point-to-point wiring. However, I wanted to do this as a PCB (Printed Circuit Board) so that I could make it both more robust and more functional. To be honest, my point-to-point soldering abilities aren't as good as they used to be; the parts keep getting smaller each year and my hands seem to get bigger.

This board has — in addition to the 24 channels of ADC — eight digital I/O ports, an extra SPI port, an I2C port, and a pair of COM2 serial ports for added expansion. In other words, this is pretty much a general I/O board. I decided that, since the last PCB I created was over 25 years ago, I would use layout software (Eagle) and send it out to Olimex



to be created. I strongly recommend Olimex for prototype boards. Once I learned how to use Eagle, it was fairly simple. Tto be honest, though, while I fixed up the current files, the originals had a few silk screen problems and one of the connectors was in the wrong order. The board was still workable, however. I strongly recommend creating PCBs for any non-trivial project.



The I/O board.

The SPI Protocol

The SPI protocol is a synchronous, simultaneously bi-directional, master-slave protocol. This means that data is clocked and transmitted from both master and slave at the same time. By definition, all communication is initiated by the master. To distinguish the active slave, the slave's chip select line is brought low. This means that one chip select pin must be allocated for each slave, unless there is only one, in which case the chip select is not needed. The master must only communicate with a single slave at a time.

Being a synchronous protocol, there is a clock signal involved. The signals involved in SPI are:

1. Chip Select (low when selected)
2. SPI Clock
3. SPI Master Out Slave In (MOSI)
4. SPI Master In Slave Out (MISO)

You need one chip select for each SPI slave in your network. If a slave is not selected, then the SPI pins for that device are floating, so they don't load down the system. If you have only a single slave on your SPI bus, you can tie the chip select line down and not worry about it.

Since the SPI bus is a single resource, you have to make sure that it is only accessed by one thing at a time.

Many different types of chips and sensors communicate via SPI. In addition to the MCP3208s, many other chips — including some EEPROM and video imagers — use SPI.

SPI is a very simple protocol. The following steps will occur during communication:

1. The master pulls the chip select down.
2. The master starts the clock.
3. The master sends data at a clock transition using the MOSI line.
4. The slave sends data (simultaneously) using the MISO line.

Typically data is sent in octets (bytes), but it is possible to send data in other lengths. Data is sent most significant bit first. Most microcontrollers think of the transmission as being done in complete bytes; however, it is important to remember that SPI is really a serial protocol where the data is transmitted one bit at a time in any length.

It is important that there is no true separate read and write with SPI. Writing a byte automatically receives a byte and vise versa. It is possible to write a byte and just ignore what is returned.

SPI can be built into hardware easily because each data transfer corresponds to the data available at a simple transition of the clock.

Because of the nature of the world, there are several allowable variants of SPI. These all involve how the clock interacts with the data streams. The data could be active on either transition of the clock and the clock could either be active high or active low.

The major disadvantage to SPI is that each slave needs a dedicated chip select pin. On the other hand, I've never run into a situation where that was a problem.

Some of its advantages include speed and simplicity. There is no addressing overhead in the protocol and — when it is necessary to both send and receive data — there is no overhead there, either. Imagine using a video processor chip that processes one frame at a time. The master could send the next frame while simultaneously receiving the previously processed frame.

SPI and the MCP3208

The MCP3208 uses SPI bi-directionally.

1. The master sends a byte consisting of five bits of information, followed by two 0s (actually these two bits don't matter).
 a. The first bit is a 1 for a start.
 b. The second bit is a 1 for single-ended conversion and a 0 for differential conversion.
 c. The next three bits are the binary number of the channel (0-7).
2. After the two bit wait, the MCP3208 sends 12 bits of data.
3. It repeats this process until finished.

The address is composed of one bit to choose between single-ended or double-ended operation, followed by three bits for the ADC channel (there are eight per chip).

Because most microcontrollers like to handle serial data

The AJ-100

The Ajile AJ-100 processor is a microcontroller that runs Java bytecodes as its machine language. The Systronix JStik is a JSimm board that has an AJ-100, 8 MB Flash, and 4 MB RAM. The board runs at a maximum of 103 MHz with 3.3 V at less than 1/2 amp. All of this is on a form factor of about 3" by 2.5." The JStik has the JSimm connector and Ethernet port, two serial ports, and a high speed I/O (HSIO) port. For this article, I'm only accessing the JStik through the JSimm bus.

in bytes, we have to prepare our bytes carefully so that everything works.

If the channel is XYZ (where XYZ is a binary number between 000 and 111) and we are doing single-ended conversion, then — to make everything work out on even bytes — we have to prepare two bytes like the following:

Byte 1: 0000011X
Byte 2: YZ000000
Byte 3: 0

Since SPI is a bi-directional protocol serial, the MCP3208 sends the results while byte two is being sent. Basically, as soon as the first two bits of byte two are sent, the MCP3208 knows which channel is being accessed. It takes two cycles of the SPI clock to start the conversion, so the data is sent back, starting with bit four of the returned data while byte two is being sent! So, it goes something like this:

1. The controller sends byte one and ignores the byte received.
2. The controller sends byte two and stores the received byte as the four highest bits of the result.
3. The controller sends a 0 byte and stores the received byte as the low-order byte of the result.

Note: I have said byte three should be 0. This is the easiest case, but it is possible to use a byte like byte one to read the same or a different ADC channel of the ADC. In this case, the host might send:

byte 1a, byte 2a, byte 1b, byte 2b, byte 1c, byte 2c ...

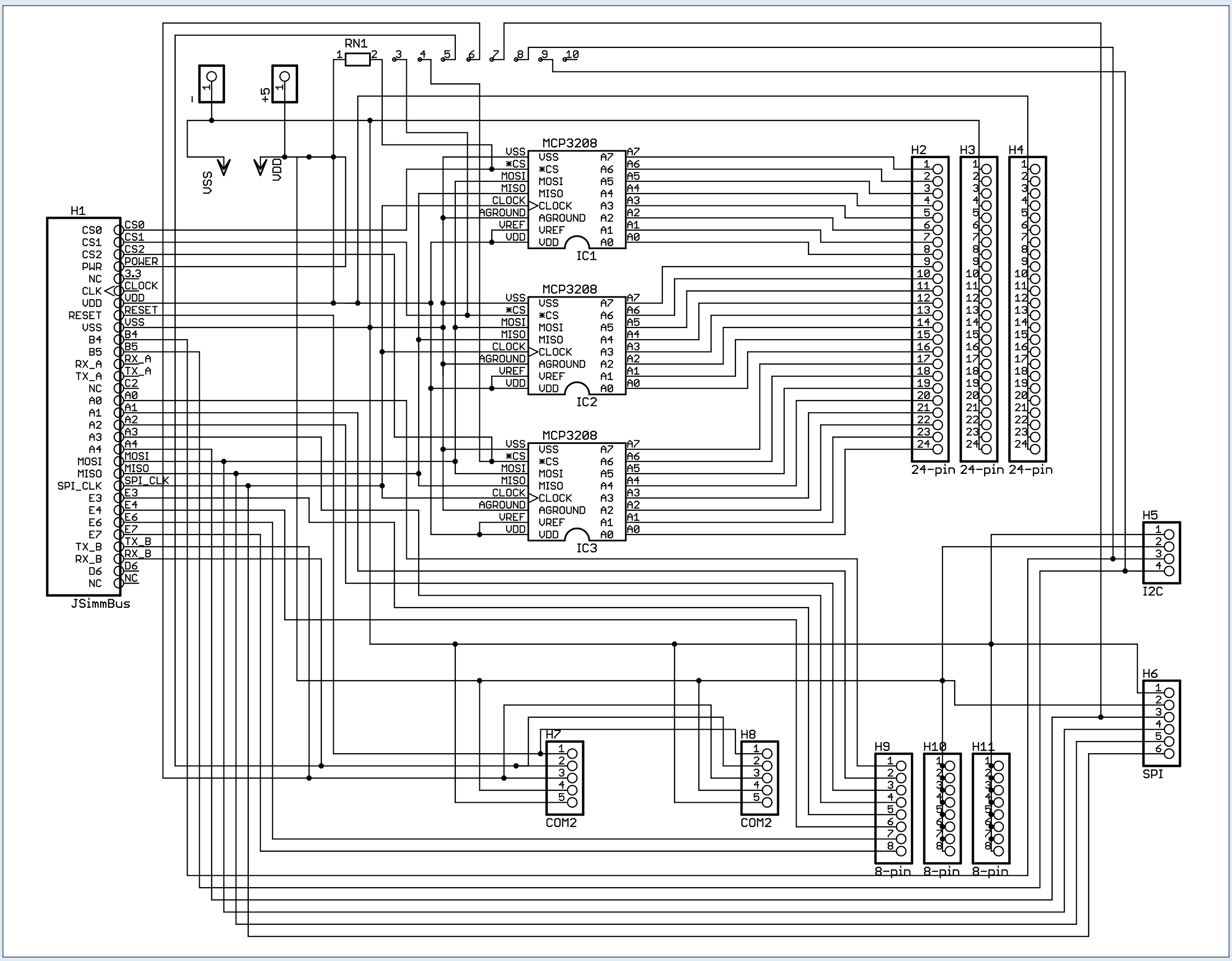
While the received stream might be:

0, high-byte a, low-byte a, high-byte b, low-byte b, high-byte c, low-byte c ...

The Software

The Ajile processors — like many other controllers — have

The I/O board schematic.



an SPI module built in. This is accessed via the SpiMaster class. One important thing to consider is that the SPI bus is a limited resource and can only be accessed by one system at a time. Therefore, the SpiMaster class is a singleton.

I originally developed the code to read this ADC as part of my open source robotics framework (The Enerd Robotics Framework). The purpose of the code in my framework is to be part of a sensor package where the user can plug in any supported sensor type and input type. The code I will use here for demonstration purposes will be a simplified version of the code in the framework. If you want to see the full code, you may look it up on my website (see Resources).

The basics of the code are as follows:

1. Wait for the SPI bus to be free and then grab it.
2. Calculate the first byte and send it
3. Calculate the second byte and send it, while reading the first byte of the answer.
4. Read the next byte.
5. Release the SPI bus.
6. Choose the correct 12 bits from the two bytes we read.

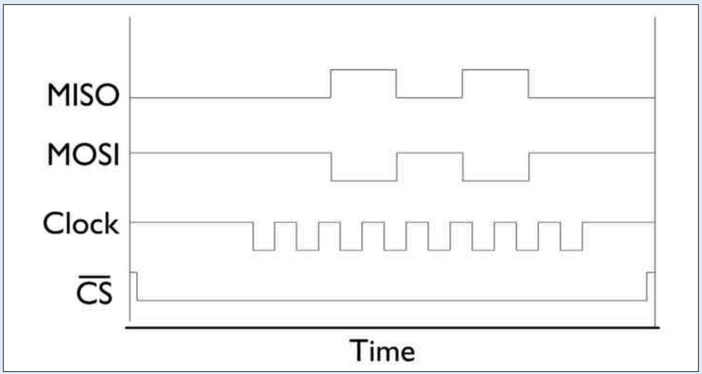
SPI and the Ajile Processors

Both the JStik (aj-100) and the JStamp (aj-80) handle SPI in exactly the same way. When programming in JemBuilder, you have to include the SPI driver and tell the SPI driver that you want to use output pins for the chip selects. For some reason, the Ajile processors only support three chip-select pins. It is easy to change this with a wrapper class that I will show you.

I have abstracted most of the SPI processing into my own SPI class. This wrapper class can easily be rewritten for use with other Java processors. The basic interface is as follows:

```
package ws.enerd.util;

public abstract class SPI  {
  public static final int BITS_8 = 8;

  public static final int DIVIDER_8 = 8;
  public static final int DIVIDER_16 = 16;
  public static final int DIVIDER_32 = 32;
  public static final int DIVIDER_64 = 64;

  protected int chipSelect = 0;
  protected int clockDivider = 0;
  protected boolean phase = true;
  protected boolean clockInverted = false;

  protected static Object[] chipSelects = new Object[16];
  protected static int maxIndex = -1;

  // Constructor
  public SPI(int csIndex, int clockDivider, boolean phase,
        boolean clockInverted)  {
    this.chipSelect = csIndex;
    this.clockDivider = clockDivider;
    this.phase = phase;
    this.clockInverted = clockInverted;
  }

  public static int registerChipSelect(Object pin)  {
    maxIndex++;
    chipSelects[maxIndex] = pin;

    return maxIndex;
  }

  public abstract boolean open(boolean wait);

  public abstract boolean open();

  public abstract void close();

  public abstract int readData();

  public abstract void writeData(int b);

  public abstract int writeReadData(int b);
}
```

SPI bus signal timing diagram.



This is an abstract class that only handles the static methods of adding chip select pins to an internal array and assigning them integer values. I make no assumptions about how the SPI communication actually works. This way, I can use the same SPI object for different Java processors.

Another object would return the correct SPI class for a given application. In my robotics framework, I have a Robot class that is designed for just this purpose.

The code to read one channel and display it on the console is even easier:

```
// Copyright 2003 by D. Jay Newman; All rights reserved
// This file is distributed under the LGPL

/**
 * Test.java
 *
 * An application to show how to read sensors via the
 * SPI interface to an MCP3208
 *
 * @author D. Jay Newman
 **/
import ws.enerd.ajile.SPI;
import com.ajile.drivers.gpio.GpioPin;

public class Test  {
  private static SPI spi = null;
  private static int chipSelect = 0;
  private static int channel = 7;

  private static int byte1 = 0;
  private static int byte2 = 0;
```

```
  public static void main(String[] args)  {

    SPI.registerChipSelect(new GpioPin
                        (GpioPin.GPIOC_BIT0));
    spi = new SPI(chipSelect, SPI.DIVIDER_64,
                        true, true);

    byte1 = ((channel >> 2) | 0x06);
    byte2 = ((channel << 6) & 0xFF);

    System.out.println("About to enter while loop");

    while (true)  {
      try  {
        System.out.println("Reading = " + readSensor());

        // We need the try to be able to sleep
        Thread.sleep(1000);
      }
      catch (Exception ex) {
        ex.printStackTrace();
      }
    }
  }

  public static int readSensor()  {
    spi.open(true);

    int result = 0;
    int b;

    spi.writeData(byte1);
    b = spi.writeReadData(byte2);

    result = ((b & 0x0F) << 8);
    b = spi.writeReadData(0);

    result |= (b & 0xFF);

    spi.close();

    return result;
  }
}
```

The output of the above code looks a bit like:

```
[TEXTIO.0]->Reading = 360
[TEXTIO.0]->Reading = 357
[TEXTIO.0]->Reading = 359
[TEXTIO.0]->Reading = 362
[TEXTIO.0]->Reading = 341
[TEXTIO.0]->Reading = 1984
[TEXTIO.0]->Reading = 1961
[TEXTIO.0]->Reading = 1976
```

The Board

Since this project was going to be part of a working robot, I wanted something more robust than just point-to-point soldering. So I downloaded the Eagle PCB design program and designed my first PCB in 30 years. Then, I sent the design to Olimex for fabrication, and got it back a few weeks later. (Next time, I may spend a bit more on postage to get it back faster; I'm not the most patient person in the world.) I'll admit, the prototype had a few mistakes (my fault), such as some misplaced silk-screening and a couple of upside-down headers. None of these were fatal errors but the board design on the *SERVO* website (**www.servomagazine.com**) is corrected.

I'm only talking about the ADC portion of the board. I also brought out eight general I/O pins, an extra SPI port, an I2C port, and a couple of COM2 ports put out by Kronos Robotics used for small coprocessor boards (basically preprogrammed PICS).

There are no active components on this board, except for the MCP3208s. The only passive component is a resistor array for pull-ups. The major expense in building this board was getting all the headers.

You'll also note that I made power and ground lines very wide. I did this because all the sensors are able to be powered from the ADC headers (power, ground, and signal). This board should be able to handle up to 2 amps on the power connectors. I use a regulated 5 volt, 3 amp power supply to provide the power to this board and the JStik connected to it. I haven't had a problem.

I usually use a second power supply for the motors. If I were remaking this board, the biggest thing I'd have is an optional second supply for any servos. Since I'm using a servo coprocessor board to handle this, I don't have to worry about it on this board.

My Conclusions

Well, the board works to my satisfaction. However, if I were redoing the board, I would make a few changes.

- I would only use two MCP3208s (16 channels total) because I haven't had to use the full 24 yet.
- I would install the coprocessor chips directly on the board rather than use a daughter-board.
- I would add another SPI port, maybe two more. SV

Resources

JStik — **www.systronix.com/**
Olimex — **www.olimex.com/**
CadSoft (EAGLE PCB editor) — **www.cadsoft.de/**
Microchip — **www.microchip.com/**
Ajile — **www.ajile.com/**
Kronos Robotics — **www.kronosrobotics.com/**
Jay's robotics website — **http://enerd.ws/robots/**

About the Author

During the day, D. Jay Newman works with a group called TLC at Penn State creating programs to aid faculty in using technology in the classroom. In the evenings and on weekends, he is busy constructing robots, writing, and — when he has the spare time — goofing off. He is old enough to have built his first computer around a 6502 with 18K of RAM when that was a large computer. The JStik that is the brain of his current robot is many times more powerful than this first computer. He lives with a wife, a dog, and an annoying parrot. He enjoys writing about himself in the third person.

Robytes

by Dave Calkins

Another month, another collection of robot trivia to amuse your coworkers and annoy your pub-mates. Surely there are more fun stories out there. Got a good story on robots? Email me: news@robotics-society.org If you'd like to get even more robot news delivered to your in-box (no spam, just robo-news) drop a line: subscribe@robotics-society.org

— David Calkins

DARPA Could Learn a Thing or Two From These Kids

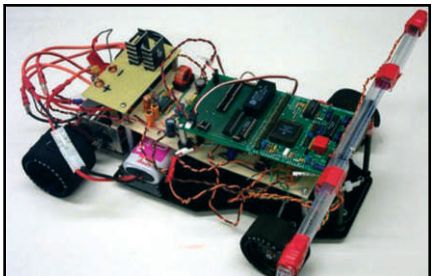


Photo courtesy of NATCAR.

Everyone remembers the DARPA challenge — making a vehicle autonomously drive from Barstow to Vegas (shades of Hunter S. Thompson ...). Sure, CMU got the furthest, but they also spent around $5 million to do so (even though, according to the original challenge rules, no current DAPRA grant recipients were supposed to compete.)

Well, in universities throughout California, students compete in NATCAR — an autonomous vehicle competition sponsored by National Semiconductor. With an emphasis on learning design circuitry, engineers modify off-the-shelf R/C cars to follow a complex line (including intersections and curves). The cars must follow the line autonomously, without turning at intersections, and finish the course in the fastest time. These cars don't move slowly either — they rip along as fast as most unmodified R/C cars and still manage to stay on the course. As the years go on, the competition will get more robust and I can foresee the lessons learned being installed into real vehicles to help drivers navigate and reduce accidents.

Which will help my editor after one of his famous three martini lunches ...

The Next Step in Combat Robots?

Photo courtesy of Greg Brotherton.

Form follows function ever after. Yeah, it's a cliché, but since when have I avoided clichés? Down in Southern California, artist/machinist/sculptor Greg Brotherton is building android sculptures that make C-3PO look like he was made from Lincoln Logs. One of his top-of-the-line models is the Mercury 5000: "Speed and stealth are key when it comes to battlefield recon. The Mercury 5000 is fast, smart, and practically invisible in action. Employing the latest advances in quantum feedback tacking, we've designed a machine that moves so fast it works outside of the timestream. Send and receive messages before they're generated. Find out where the enemy is going before they know themselves!

"It's all possible with power to spare from the 5000's QT-Atomic reactor encased in a sleek and powerful body. A massive piece of articulated steel, Mercury stands nine feet tall and weighs around 400 pounds. The hand-polished and intricately welded steel skin is coated inside with a rust blocking agent that is used on oil tankers; the outside has a tough, clear marine varnish. The base is a torch-cut steel plate, bolted to inch-thick plywood with casters."

Okay, it doesn't really have an atomic reactor in it, but Anthony Daniels doesn't really speak 10 million languages, either.

P.S. This would look *great* next to my Christmas Tree.

Robots — They'll Even Do Street Art!

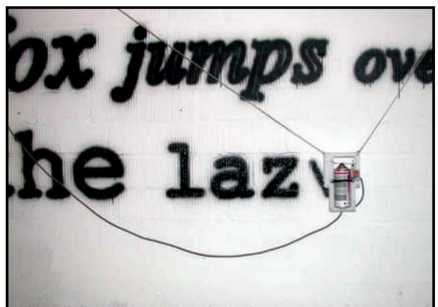


Photo courtesy of www.hektor.ch

This one gets filed in the "I thought I'd seen it all" section.

Bored by having written software for several years, Swiss student Jürg Lehni wanted to make something tangible. So, he called his pal Uli Franke and, together, they decided to make a robot that ... wait for it ... did graffiti — like on walls. The stuff that annoys high school principals and civil servants alike.

Using two stepper-motors,

toothed-belts, and a small circuit board, the robot hangs from a wall and paints its way to immortality. Now, don't just think that it's limited to simple ASCII text. Nope, these guys went all the way. Using his own software — Scriptographer — the robot takes any Adobe Illustrator vector file and paints it on the wall. Snide comments, cool artwork, avant-garde drawings, photos — just about anything. With this method, civil disobedients can safely blame the robot for their graffiti *and* make sure everything is spelled right in advance ...

Can the rhetorical bank robbing bot be far behind?

Yet Another Way Robots Help Save Lives

We all know that robots are slowly making their way to the battlefield. iRobot now makes robots that can go safely where humans cannot — and can be the first ones shot — to alert humans as to where the bad guys might be. However, robots will never be able to serve alone on the field of battle and humans will always get hurt. Often, when medics are trying to evacuate the wounded, the medics themselves end up being shot.

Applied Perception, Inc., of Pennsylvania and Remotec of Tennessee have received a $1 million grant to develop a robot team that will help in retrieving wounded soldiers without exposing more humans to enemy fire. The robot team will be comprised of a small robot that rides inside a bot that is about the size of a compact car. The smaller robot leaves the larger robot to get the wounded and brings them into the larger bot for evacuation. The larger bot is capable of moving up to two wounded people to a safer place or a field hospital at a time. Just as long as the bots don't drink as many martinis as Hawkeye and BJ, our troops should be fine.

C-3PO Catches Up to R2-D2

Photo courtesy of ©Lucasfilm Ltd. & TM. All Rights Reserved.

If you recall from the January column, R2-D2 made it into the Robot Hall of Fame as an inaugural awardee. Well, this year's bots have been named and C-3PO finally got his due. Right next to him will be the closest thing to a functional C-3PO — the Honda Asimo. Every year, the judges pick both real and fictional robots to include in Carnegie Mellon's Hall of Fame.

This year had five inductees. In addition to the above, my all-time favorite Sci Fi robot got in: Robby the Robot — from *Forbidden Planet* and *Invisible Boy*. (No, he was not in *Lost in Space*. That was B9.) The most influential robot in Japan — Astroboy — also made it in on the Sci Fi side. Closing out the inductees was SRI's Shakey the Robot, one of the world's first sensing, mobile robots. Shakey could do range-finding, had a video camera and bump sensors, and got around pretty well for a robot built in 1966 — although he was a bit shakey in its movement (hence the name).

Don't forget to send each one a congratulations telegram! **SV**

EVENTS CALENDAR

Send updates, new listings, corrections, complaints, and suggestions to: **steve@ncc.com** or FAX **972-404-0269**

Just a few days before I prepared this month's event update, DARPA announced the date for the next Grand Challenge: October 8, 2005. They also announced that the prize had been doubled. The winner will now receive $2 million.

One special competition to look for in September is the SRS/*SERVO Magazine* Robo-Magellan event at Robothon in Seattle, WA. Autonomous robots will be traveling an outdoor course over a varied terrain that includes both natural and man-made obstacles. It's almost like a miniature version of the Grand Challenge.

— R. Steven Rainwater

For last minute updates and changes, you can always find the most recent version of the complete Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

(No confirmed August events)

September

3-6 **Dragon*Con Robot Battles**
Atlanta, GA
Radio-controlled vehicles destroy each other at a famous science fiction convention.
www.dragoncon.org/

6-7 **RoboCup Junior Australia**
Queensland, Australia
There are over 600 RoboCup Junior teams in Australia. Regionals narrow this number down to about 200 teams that will compete at the University of Queensland to see who's the best at building LEGO-based, autonomous soccer robots.
www.robocupjunior.org.au/

11 **ABU Robocon**
Seoul, Korea
Autonomous robots must build a bridge and then move objects across it.
www.kbs.co.kr/aburobocon2004/

25-26 **Robothon**
Seattle Center, Seattle, WA
At this Seattle Robotics Society event, people from all around the world come together to present new robotic technologies, show off their special robotic creations, and compete in several robotic competitions and activities. The Robo-Magellan competition is sponsored by *SERVO Magazine*.
www.robothon.org/

October

8-10 **Robot Fighting League National**
Herbst Pavilion, Fort Mason Center
San Francisco, CA
Radio-controlled vehicles destroy each other in San Francisco.
www.botleague.com/

9-10 **RoboMaxx**
Grants Pass, OR
Includes a range of events for autonomous robots, including maze solving, 3 kg sumo, mini sumo, micro sumo, and nano sumo.
www.sorobotics.org/RoboMaxx/

21-23 **Tetsujin**
RoboNexus, Santa Clara, CA
SERVO Magazine's weight lifting competition for powered, articulated exoskeletons offers an event incorporating the technology of the future. The event is being held in conjunction with RoboNexus. See page 4 of this issue for more information or visit the website for rules and full details.
www.servomagazine.com/tetsujin2004/

November

22-24 Critter Crunch
MileHicon, Marriott Southeast, Denver, CO
The Denver Area Mad Scientists were pitting autonomous and remote-controlled robots against each other long before commercial events like "BattleBots" and "Robot Wars."
www.milehicon.org/

27-31 FIRA Robot World Cup
BEXCO, Busan, Korea
Check out all the usual categories of robot soccer, including humanoid, single, team, khepera, and many others. Visit the website for further details.
www.fira.net/

November

6 CIRC Autonomous Robot Sumo Competition
Peoria, IL
In addition to sumo, this year's event includes some R/C combat events.
www.circ.mtco.com/competitions/2004/menu.htm

13-14 Eastern Canadian Robot Games
Ontario Science Centre, Ontario, Canada
Includes BEAM events, including autonomous sumo and a fire fighting competition.
www.robotgames.ca/

22 Texas BEST Competition
Reed Arena, Texas A & M University
College Station, TX
This is the big one, where the winners from the regionals compete.
www.texasbest.org/

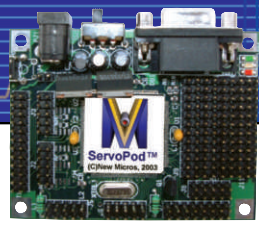
26-27 War-Bots Xtreme
Saskatoon Saskatchewan, Canada
Robots (R/C vehicles) attempt to destroy each other to win $10,000.00 in prize money.
www.warbotsxtreme.com/

SERVO COPROCESSOR BOARDS

PRODUCT	Channels	Controllers per serial bus	Servos per serial bus	Serial Connection	Serial Type	Baud Rate	Automatic Baud Rate Detection	Pulse Width Range	Pulse Resolution per 180 degrees
Parallax Servo Controller Parallax, Inc. www.parallax.com	16	2	32	Two wire (Sig, GND)	RS232, TTL Level RS232	2400, 38400	No	500-2,500 µs	1,000
Mini SSC II Scott Edwards Electronics, Inc. www.seetron.com	8	16	255	Two wire or RJ45 Phone jack	RS232, TTL Level RS232	2400, 9600	No	500-2,500 µs	254
SV203C Pontech www.pontech.com	8	257	2,040	Three wire (Tx, Rx, GND)	RS232, TTL RS232, SPI, Sony IR Remote	2400-19200	No	520-2,550 µs	254
Servo 8T BasicX - Net Media, Inc. www.basicx.com	8	8	64	Three wire (Tx, Rx, GND)	RS232 & TTL Level RS232	2400-19200	No	480-2,520 µs	254
Serial 8 Servo Controller Pololu Corporation www.pololu.com	8	16	128	Two wire (Sig, GND)	RS232 & TTL Level RS232	1200-38400	Yes	250-2,750	5,000
Serial 16 Servo Controller Pololu Corporation www.pololu.com	16	8	128	Two wire (Sig, GND)	RS232 & TTL Level RS232	1200-38400	Yes	250-2,750	5,000
Digital Servo Controller Ohmark Electronics www.ohmark.co.nz	8	4	32	Two wire (Sig, GND)	RS232 & TTL Level RS232	2400, 9600	No	500-2,500 µs	255
SSC-12 Lynxmotion, Inc. www.lynxmotion.com	12	1	12	Two wire (Sig, GND)	RS232	9600	No	500-2,500 µs	254
Quad Serial Servo Controller Phigets USA www.phigetsusa.com	4	115	460	Four wire (D+, D-, VDD, GND)	USB	1.5 Mbps	n/a	0-2,500 µs	2,000
USB Servo 2 Controller Ackerman Computer Sciences www.acscontrol.com	8	1	8	Four wire (D+, D-, VDD, GND)	USB	n/a	n/a	500-2,500 µs	1,000
ServoPod - USB New Micros, Inc. www.newmicros.com	26	unlimited	unlimited	Depends on serial interface	RS232, TTL Level RS232, RS 485, SPI, CAN	600-115200, 1 MHz CAN, 20 MHz SPI	No	4 ns - 13 ms	32,768
Mini Atom Bot Board Lynxmotion, Inc. www.lynxmotion.com	20	1	20	Three wire (Tx, Rx, GND)	TTL Level RS232	300-38400	No	400-2,600	254
SD20 Devantech-Acroname www.acroname.com	20	1	20	Four wire (5 V, GND, SCL, SDA)	I2C	up to 100 kHz	No	400-2,600 µs	255
SD21 Devantech www.robotelectronics.co.uk	20	1	20	Three wire (GND, SCL, SDA)	I2C	up to 100 kHz	No	10-65,000 µs	1 µs
PicoPic PicoBytes, Inc. www.picobytes.com	20	1	20	Two wire (Sig, GND)	RS232, TTL Level RS232	1200-115200	No	500-2,400 µs	1 µs
Servio PicoBytes, Inc. www.picobytes.com	20	1	20	Four wire (Tx, Rx, 5 V, GND)	TTL Level RS232, RS485	1200-115200	No	500-2,400 µs	1 µs

MATRIX

by Pete Miles

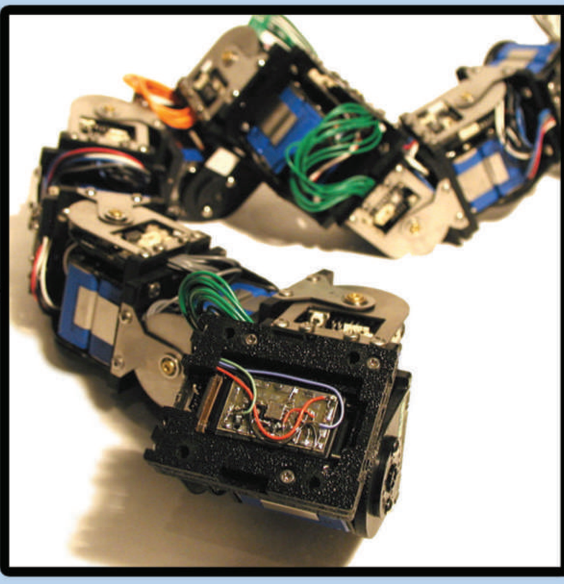


Look for a new comparison matrix every month! Upcoming topics include oscilloscopes, batteries, drive motors, and radio data links. Pete is a busy guy, so — if you're a manufacturer of one of these items — please contact him in advance with your product information: BrainMatrix@servomagazine.com

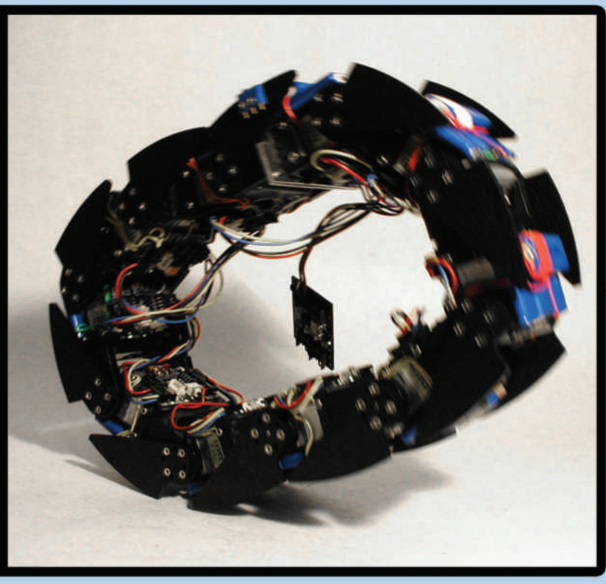
Position Update Frequency, Hz	Velocity Control	Velocity Resolution, Steps per update period	Acceleration Control	Acceleration Range	A/D Inputs	Digital Inputs/Outputs	EEPROM/RAM (size)	Number of Commands	Servo Position Feedback	Firmware Upgradeable	Programmable	Size (inches)	Controller Power Requirements	List Price (in US dollars)
50	Yes	0.75 to 60 seconds for full range of motion	n/a	n/a	0	1/0	No	8	Yes	Yes	No	2.7 x 2.8	5 VDC	39.00
60	No	n/a	No	n/a	0	0/0	No	2	No	No	No	1.4 x 2.1	7-15 VDC 10 ma	44.00
50-70	No	n/a	No	n/a	5	8	Yes (8 KB)	16	Yes	No	Yes	1.4 x 2.2	7-12 VDC	85.00
65-79	No	n/a	No	n/a	0	0/0	Yes (128 B)	11	Yes	Yes	No	2.5 x 3.0	5.5-9 VDC 15 ma	39.95
50	Yes	128	No	n/a	0	0/0	No	6	No	No	No	1.45 x 1.7	5.6-25 VDC 15 ma	32.00
50	Yes	128	No	n/a	0	0/0	No	6	No	No	No	1.5 x 2.3	5.6-25 VDC 15 ma	52.00
50	Yes	254	No	n/a	0	0/0	No	13	No	No	No	1.6 x 1.5	4.6-11 VDC	35.75
50	Yes	16	No	n/a	0	0/0	No	4	No	No	No	1.8 x 2.0	9.6 VDC Max, BEC	59.95
50	No	n/a	No	n/a	0	0/0	No	2	No	No	No	2.1 x 1.7	6-12 VDC	56.00
40	Yes	255	No	n/a	0	8/8	Yes (16 B)	7	No	Yes	No	2.6 x 5.6	5-6 VDC	94.95
38-76	Yes	32,768	Yes	32,768	16	16/26	Yes (204 KB)	Depends on programming language	Yes	Yes	Yes	2.0 x 3.0	6-12 VDC 20 ma	199.00
50	Yes	254	No	n/a	4	20/20	Yes (32 KB)	4	No	Yes	Yes	2.3 x 3.0	6-12 VDC 20 ma	84.90
50	No	n/a	No	n/a	0	0/0	No	2	No	No	No	28 pin IC	5 VDC 5 ma	16.50
50	Yes	255	No	n/a	0	1/0	No	3	No	Yes	No	1.7 x 2.3	6-12 VDC 10 ma	—
30	Yes	255	No	n/a	0	20/20	No	4	No	Yes	No	1.5 x 2.5	6-30 VDC	49.95
30	Yes	255	No	n/a	8	20/20	Yes (256 B)	30	Yes	Yes	Yes	2.5 x 2.5	6-30 VDC	99.95

PARC PolyBots

by Dan Danknick



The advanced G1v5 (generation 1, version 5) module in a 3-D linear form.



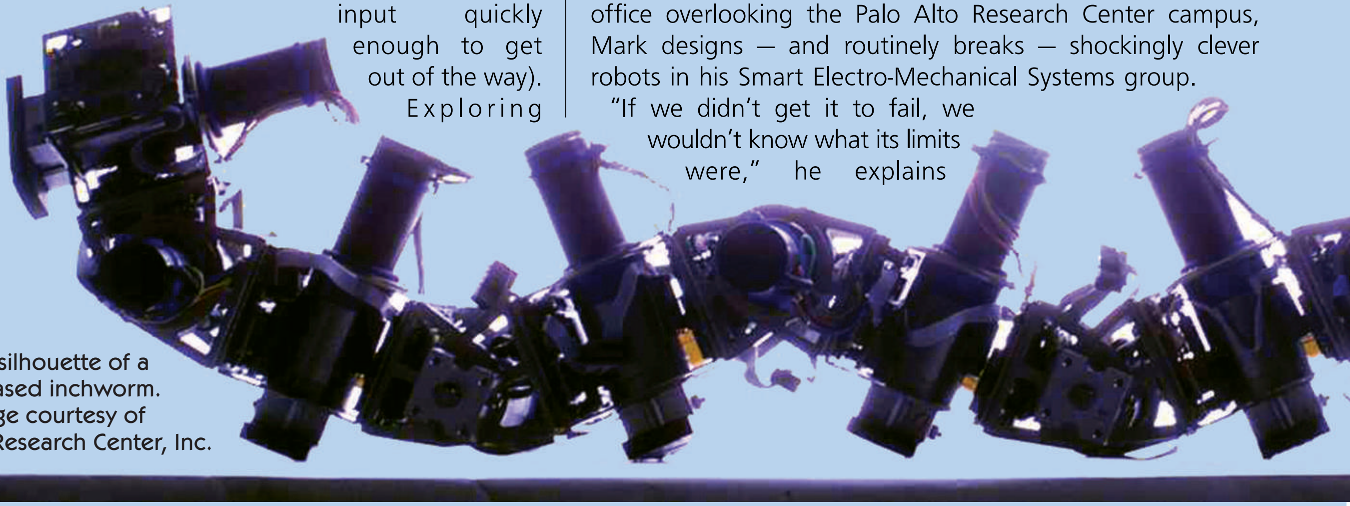
G1v4 modules in a ring show the "rolling loop" gait in action.

Everyone who has ever constructed a robot has been exposed to the "can do" domain: for a given design, there is a field of things a robot can do, bounded by a set of limits where it simply fails to function. The vectors of this field may be physical (can't climb stairs due to small wheels), electronic (can't generate enough torque to get up a ramp), or computational (can't respond to sensor input quickly enough to get out of the way). Exploring these limits usually causes a robot engineer to consider upgrades — or perhaps the next complete incarnation. What if the design of a robot was so fundamental that the boundaries of the "can do" domain were practically erased?

Bio-inspiration is good; mimicry is

Meet Mark Yim, senior SEMS researcher at PARC. Equipped with a Ph.D. and a radical machine shop near his office overlooking the Palo Alto Research Center campus, Mark designs — and routinely breaks — shockingly clever robots in his Smart Electro-Mechanical Systems group.

"If we didn't get it to fail, we wouldn't know what its limits were," he explains



The silhouette of a G2-based inchworm. Image courtesy of Palo Alto Research Center, Inc.

while flanked by the wreckage of various robotic efforts.

Polymorphism

Mark builds modular, self-reconfiguring robots, which have been dubbed "PolyBots." They are large robots — made of smaller robots — and can change shape on the fly. They have a number of advantages over application-specific robots:

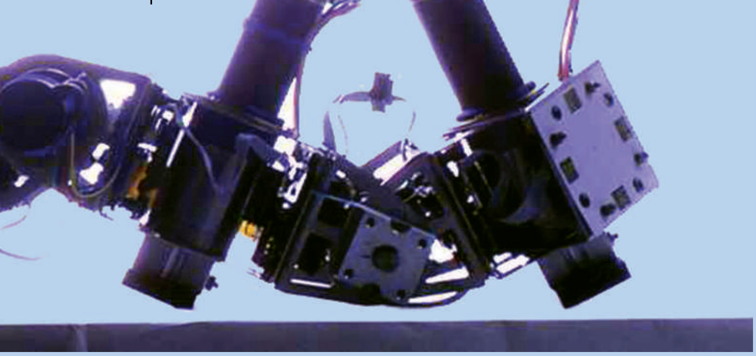
- They are versatile and can adapt to dynamic tasks.
- They are robust in that they are redundant.
- They are low in cost due to economy of scale.

Each module of the PolyBot is a largely self-contained unit comprised of a power source, CPU, sensors, and mechanics. In the seven years Mark has been working at PARC, the module design has steadily advanced — from a G1 (first generation) effort using hobby R/C servos and requiring manual reconfiguration — to the G5, which uses a brushless motor, high torque planetary gearbox, CANbus communications, PowerPC CPU, and an SMA-based interconnect latch to reduce the number of moving parts.

not recommended.

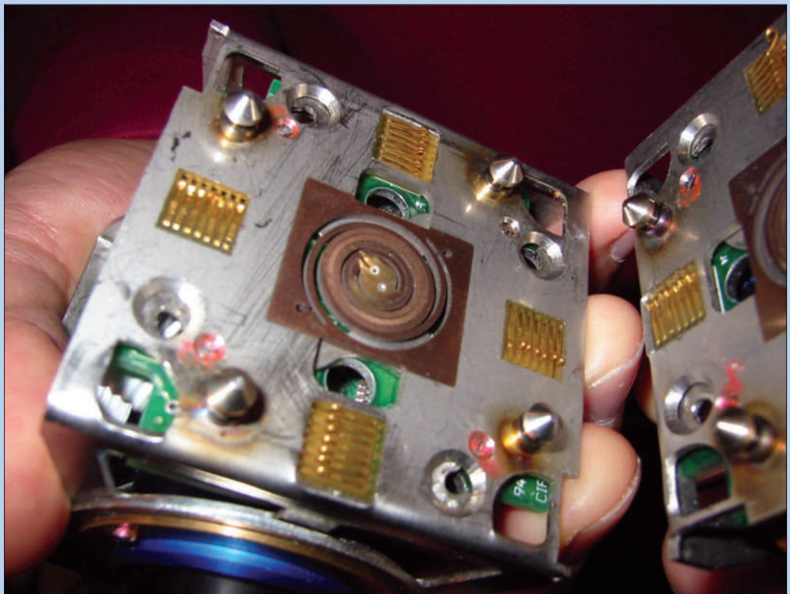
Detailed information on the transitional generations is available on the PARC website (listed in the links sidebar).

A fundamental feature of each module is that it bends in the middle, effectively shifting its mating surfaces in space. When a series of modules are connected, this creates a change in the geometry of the entire robot and permits crawling, rolling, and even walking — if the composite shape admits such motion. There is little question that this radical approach to robot building quickly becomes fantastically complex to design and control, but the power of a self-reconfiguring robot is equally fantastic.



Other advantages of the modular design are less obvious, yet quite important. For example, if one module fails, it can be left behind (or, more likely, it will choose to disconnect itself once it no longer has anything to add). If a PolyBot is in motion, it can monitor areas of interest "on the fly" by simply using the sensors in the closest module. (This is like running between the cars of a moving train in order to observe a tree on the side of the tracks.)

A G1v5-based snake inches its way into a hole.



A close-up of the G2 mating surface, showing alignment pins and electrical interconnect fingers. The swirl at the center is an SMA spring that releases the alignment pin latch.

Modules can even reconfigure themselves to form different PolyBot structures, conferring alternate gaits to the robot in the middle of a mission. For example, a linear "inchworm" could latch its ends into a ring to make rolling downhill more efficient. This is certainly high in "cool factor," but it is a serious software challenge. How should the distributed control software detect a downhill slope, much less when it ends? Forming a useful software abstraction of the mechanical system takes even more effort than machining all of those tiny bearings and frames.

Researcher Mark Yim shows a walking biped made of G1v5 modules.

Canned or Fresh?

Once you have assembled all of the parts of a self-reconfiguring robot, you have to decide how to make it move. There are two approaches to choosing the "motion profile" that the robot will use: canned or adaptive.

Canned motions are driven from a "gait control table" in a database, where each cell encodes the desired joint angle against time. Traversing the cells to command the servos is like playing a movie: You don't see the individual frames, just the final motion.

Adaptive motion is much more complex and is usually represented as a state machine, where each state is a particular canned motion. As sensors read the environment, the control program moves between states and the robot exhibits different motions. Sensor readings can also modify parameters of the canned motion within a state. For example, if a PolyBot discovers that it is not making any actual headway — even though it is executing a gait — it may choose to slow down and see if traction can be improved.

Polynomic Complexity

The modular approach is a serious software and processing challenge, especially when geometric decisions are made. "Reconfiguration planning is an NP-hard problem," explains Mark. NP-hard is a computer term used to describe algorithmic complexity that increases exponentially with the items involved. Take a four module PolyBot — in two dimensions, it can assume six unique forms. Simply increase this to eight modules and you have 500 forms. Add in that third physical dimension and things really go wild. Don't forget that there are other control dimensions beyond physical structure: joints have angular limits, motors have maximum holding torques, etc. Even a non-trivial construct of modules becomes a processing nightmare to reconfigure on the fly.

Run Like a Cheetah?

It might be tempting to take a cue

from nature and mimic animals that are already successful. So, is biology a good pattern? Mark explained that, when he first started working on PolyBots, he promised himself that he would not restrict himself to biological gaits. “It turned out that every one of my efforts mapped to something alive.” There were also a few surprises: The rolling gait of a snake is the most efficient, as real snakes push against whatever they sense, “and scaled skin may not be that important.” Mark currently advises that, “Bio-inspiration is good; mimicry is not recommended, as animals do things that robots do not.”

Give a Robot a Job

Now on their fifth generation, PolyBot modules are sticking to a cubic form from which various geometries can emerge. The most common composite assemblage under research is a linear snake. With its small cross section, a snake has an advantage in cluttered environments, which are emerging as the most likely use of PolyBot machines during search and rescue tests. “We built one for NASA that would crawl into a cave and look for acidophiles — bacteria that prefer acid.” The exploration didn’t discover any new life, but it was a great live test of the PolyBot concept. “Other projects on our bench include adapting PolyBots for deep sea mining and space research. The redundant aspect of this approach makes for innately rugged equipment. Replacement of damaged modules on the fly is the kind of thing you want in space.” **SV**

Useful Links

PARC Modular Robotics
www2.parc.com/spl/projects/modrobots/

Kasper Støy’s Research
(Self Reconfigurable Robots Using “Recruitment Gradients”)
www.mip.sdu.dk/~kaspers/

Darmouth Robotics Lab
(Reconfiguration Through Cellular Automata)
www.cs.dartmouth.edu/~robotlab/robotlab/robots/

Information Sciences Institute
(CONRO and Spider Link)
www.isi.edu/robots/conro/

The Foresight Institute
(The Ultimate Compilation of Internet Links)
www.foresight.org/Nanomedicine/Swarm.html

by James Antonakos

THE ASSEMBLY LINE

Nervous System — Brain, Spine, and Nerves

This month, I am beginning a new column that shows the step-by-step process that you can use to develop a custom robotic application. Through a mixture of hardware, software, and theory, the components of the robotic system can be designed, interfaced with each other, and integrated into a working product.

Where do we begin? At the beginning? Where might that be? For comparison, let's examine the human nervous system and see how its components and operations relate to the world of robotics.

The human nervous system consists of the brain and spinal cord (the central nervous system), as well as nerve tissue that communicates sensory and motor information between the brain and body. This nerve tissue is referred to as the peripheral nervous system. Sensory information is gathered from sources both outside and inside the body. Conscious, subconscious, and reflexive processes monitor and control all aspects of the system.

The nervous system of a robot may need to mimic all or part of a human's nervous system, depending on the functions and complexity of the operations performed by the robot. Let's examine the components of the robotic nervous system.

Perhaps the most important component is the "brain." This is the microprocessor, microcontroller, or state machine that controls all functions of the robot and provides it with its intelligence. There may even be multiple processors, each controlling a different portion of the robot. A control program is required to provide the robot with its decision making capability and other housekeeping duties, such as initialization during power on, navigation, and personality. Many useful artificial intelligence algorithms will come in handy here, as well, to assist with path finding while searching a maze, environmental reconnaissance, data and object representation, and behavior modification.

Within the brain, there is also memory — both short term and long term. Short term memory may be implemented as RAM. Long term memory may be EPROM, EEPROM, or Flash memory. Sometimes, information from short term memory is transferred to long term memory. A mechanism to determine when this transfer is necessary must be incorporated to establish learning. We may even choose to download the control program into the robot each time it is turned on.

How much brain power is needed? Surely, it would be overkill to put a 2.4 GHz Pentium 4 CPU into a robot arm designed only to paint car doors on an assembly line. Depending on the application, an eight-bit CPU may be more than enough.

Even number crunching must be considered. Are floating-point numbers required or can we get away with integers or fixed-point formats? Certainly, real time behavior will require fast calculations, so we may need to rely on one or more techniques for doing calculations quickly (dedicated hardware, lookup tables, or estimates). The human brain — with its trillions and trillions of specialized cells — has a great advantage here, for it may not be possible to put a large number of parallel processors into a machine.

What about conscious, subconscious, and reflexive behavior? Consciousness in a robot is a reflection of the continuous operation of the control program. Subconscious behavior may be implemented through background processes that run independently of the control program. Reflexive behavior may bypass the control program entirely, being directly hard-wired into the system so that cause and effect occur automatically.

Next, we'll examine the "spinal cord." This is simply a communication medium. Electronically, this is implemented as a simple parallel or serial data bus, multiple input/output ports, or even a communication network, such as the Ethernet.

Is the data continuous or intermittent? Should the input devices be polled or interrupt-driven? Will one sensor have a higher priority than the other sensors?

Now, we come to the "nerves." Like humans, robots require information from their environment. This information helps keep the organism alive, interact with other organisms,

and go about its tasks in an intelligent manner. Like humans do with the senses of touch, sight, and hearing (We'll leave taste and smell out of the discussion for now.), the robot must be capable of gathering sensory information. Fortunately, there are many components and circuits available to assist in this area. Some of these are thermistors (for sensing temperature), microphones, switches, phototransistors, image sensors, and tachometers. Analog sensors will typically require their voltage or current output to be digitized for use by the control circuitry. Thus, analog-to-digital and digital-to-analog converters may be necessary for the application.

Motion is accomplished through the use of stepper or servo motors, hydraulics, or other means. Power amplifiers may be necessary to drive the motor circuitry from the low-voltage, low-current electronic circuitry.

Some aspects of the robot's control system may be open-loop by nature, while others will require a closed-loop approach. For example, if the robot hits a wall, a short message ("ouch") may be output from a speaker or displayed on an LCD panel. This is an open-loop feature. However, controlling the speed of the robot as it travels across the ground may require a closed-loop system to constantly monitor the motor's speed and provide feedback to the control program so that adjustments can be made. Avoiding walls may require the use of ultrasonic transducers to constantly measure the distance between the robot and its surroundings.

The overall system may need to communicate with a master controller — or even with other robots in a cooperative nature. Here, we might use wireless communication (via radio frequencies or infrared) or even a simple clapping of hands together to start/stop activity.

Last, humans are unpredictable creatures. Sometimes, we do things that do not make sense. Our behavior has an element of randomness woven through it. How do we accomplish this unpredictability with hardware or software?

TERMS

Environmental Reconnaissance

Gathering data about the environment, such as the distance to walls, the amount of light, noise levels, and the number and type of objects encountered.

RAM (Random Access Memory)

Also called read/write memory, used for temporary storage. RAM forgets its data when power is turned off.

EPROM (Erasable Programmable Read-Only Memory)

Memory that may only be read after it is programmed. Used to store programs and important data. Does not forget its data when power is off.

EEPROM (Electrically Erasable PROM)

Similar to EPROM, but allows data in individual locations to be changed while power is on.

Flash Memory

Similar to EEPROM, but requires entire banks of data to be changed, rather than individual locations.

Floating-Point Number

A real number encoded into a specific binary format. Requires a floating-point unit (or time-consuming software) for calculations.

Fixed-Point Number

A real number whose representation fits into an integer format. Allows the use of fast math operations.

Lookup Table

A data table of numbers stored in memory that allows a result to be looked up instead of calculated to save time. For example, the lookup table may contain all of the sine and cosine values of angles from 0 to 90° in 1° steps.

Polled Device

A peripheral device whose status is constantly scanned (polled) by the system for activity.

Open-Loop Control System

A control system that has no feedback. The state of the system is only dependent on the inputs.

Closed-Loop Control System

A control system where the output is sampled and fed back to the input. The output has an effect on the new state of the system.

Parallel Processors

Two or more processors working on the same task. They may communicate via network messages or shared memory.

LCD (Liquid Crystal Display)

A low-power, electronic display that shows numbers, letters, and other symbols in a dot-matrix format.

SIMULATION

The robot scans the wall in front of itself. Two ultrasonic transducers measure the distance to the wall. Seeing an opening on the right, the robot adjusts the direction of its front wheels and creeps forward via its propulsion motor. The opening grows larger as the robot moves forward. Behind the opening is a bright light. The robot, through inadequate programming, confuses a lit hallway with the beacon on its wall-mounted home-base power source and speeds up, moving toward the opening faster and faster. It is eager to locate the base station, plug itself in, and recharge its battery. Then it passes through the opening and — before it can stop — tumbles down a flight of stairs and breaks into several pieces.

The technician looks away from his computer screen, where the image of the broken robot is frozen. A pop-up window appears on the screen, with a button labeled "Repeat Simulation?" The technician shakes his head and says, sadly, "Looks like I need to work on the hallway subroutine."

Simulation? Virtual robots? Now, there is an interesting way to have fun experimenting with robots without needing the skills to build real ones. A great deal of programming experience is required, though, along with tireless testing and debugging. With software routines replacing hardware components, there is no cost for a faulty design or incorrect control program, except the time spent in the development process. The attributes of the robot can be modified and a new simulation run to see how the behavior of the virtual robot changes. Through the power of real time 3-D graphics, the screen can display what the robot sees, as if there was a camera mounted on the robot or the camera can fly around the virtual robot, examining it from many different angles.

In addition to the software that may be required to operate a robot, simulation software to exercise the robotic control program may play just as important a role. What is required? A database of the virtual world where all objects (walls, lights, the robot itself, and obstacles, such as boxes, stairs, etc.) are represented, an editor for the database that provides a mechanism to create and modify the virtual world, a graphical rendering application to generate the real time view of the simulation environment, emulation of the robotic control program, and possibly even a networking component to allow distributed simulations with each computer connected to the network simulating one or more robots within the virtual world are all necessary.

Even better, if there is no control program written yet, the simulation software may allow the designer to control the virtual robot, using the mouse or keyboard to steer it, manipulate its arms or legs, or even take a tumble down the stairs — just for fun. The entire simulation can be recorded, saved, and played back later — faster or slower, forward or reverse, with different views and enough data to satisfy the number-cruncher in all of us.

Playing with robots, both real and the virtually imagined, can be a great learning experience. With the right simulation software, anyone can do it.

In upcoming "The Assembly Line" columns, we will examine the individual components of the robotic nervous system. Eventually, there will be enough pieces to create your own robot, tailored to your needs. I believe in the building-block approach to system design. Through creativity, the individual blocks may be connected in an infinite number of combinations, making each robotic application unique. **SV**

ABOUT THE AUTHOR

James Antonakos is a Professor in the Departments of Electrical Engineering Technology and Computer Studies at Broome Community College, with over 28 years of experience designing digital and analog circuitry and developing software. He is the author of numerous textbooks on microprocessors, programming, and microcomputer systems. You may reach him at antonakos_j@sunybroome.edu or visit his website at ***www.sunybroome.edu/~antonakos_j***

TETSUJIN TECH COMPETITOR PROFILE

Here is a sneak peek at the teams currently building for *SERVO Magazine's* powered exoskeletal weight lifting competition. For more information on the event, visit **www.servomagazine.com/tetsujin2004/**

The Widgets

Sanford, FL

Team Leader: Bryan Hood, student

Background: My team is made up of high school students. We are all in the International Baccalaureate program and enjoy math- and science-related competitions. I have been interested in robotics for the past few years and have been steadily learning more about the topic. The rest of the team have not been involved with robotics, but we have all been exposed to many of the fundamentals through physics and calculus. We are all eager to compete and, hopefully, will create a challenge for the other teams, as well as ourselves.

Motivation: I'm competing in this to become more involved with large robotic applications and to have a little bit of fun. It should be a great learning experience and will probably help with my college admissions.

Strategy: Keep things as simple as possible! We have eliminated some functions of the exosuit because they are not essential to walking or lifting weights. This reduced the system weight, cost, and the complexity.

Largest Obstacle: We have several factors to overcome in this project. First, we are a small team of high school students and we do not yet have a mentor. We don't have a very large budget, so we are looking for sponsors and doing a lot of fund raising.

Academic Focus: I am going to be a junior in high school and I would like to go to MIT, CMU, or Cal Tech to get a degree in engineering and mathematics.

Construction Materials: The frame of the exosuit will be fabricated from steel, aluminum, and nylon.

Power Source: Either batteries or HPA (high pressure air), depending on which type of actuator we choose.

Estimated Cost: $5,000.00 to $10,000.00

Contact: BNHrobotics@hotmail.com

Team Technotrousers

San Diego, CA

Team Leader: Donald Engh, manufacturing engineer
Team Engineer: Dan Rupert, high school teacher

Background: My teammate Dan and I are both mechanical engineers who enjoy building things — and really like gadgets! Dan has been involved in a number of robotics contests, such as FIRST and BotBall, as well as Electrathon racing. I have built working prototypes of interesting electro-mechanical devices.

Motivation: I'm competing because it looks like a fun design challenge that could have significant future applications.

Strategy: We are keeping things simple while trying to make the lightest exoskeleton possible that will still perform well in the challenge and conform to the rules.

Largest Obstacle: Our biggest challenge is time, as we both have full-time jobs.

Academic Focus: My teammate Dan attended Drexel University and majored in Mechanical Engineering, while I attended UC San Diego, majoring in Engineering Science.

Construction Materials: We are fabricating a square chromoly steel tubing frame supported by aluminum "feet."

Power Source: The exosuit will be powered by DC electric motors driving an Acme screw system. Torsion springs supplement the screw drive.

Estimated Cost: $1,000.00 in materials and parts

Contact: rupedog@hotmail.com

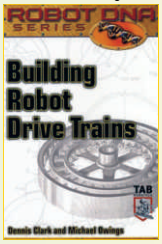
This Is One Event You Don't Want to Miss!

Tetsujin 2004 is being held during the RoboNexus International Conference and Exhibition, October 21-23, 2004. For registration and attendance information, visit **www.robonexus.com**

Mind Candy For Today's Roboticist

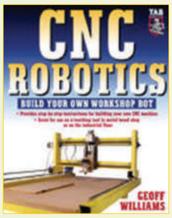
The *SERVO* Bookstore

Building Robot Drive Trains
by Dennis Clark / Michael Owings



This essential title in McGraw-Hill's *Robot DNA Series* is just what robotics hobbyists need to build an effective drive train using inexpensive, off-the-shelf parts. Leaving heavy-duty "tech speak" behind, the authors focus on the actual concepts and applications necessary to build — and understand — these critical, force-conveying systems. **$24.95**

CNC Robotics
by Geoff Williams



Now, for the first time, you can get complete directions for building a CNC workshop bot for a total cost of around $1,500.00. *CNC Robotics* gives you step-by-step, illustrated directions for designing, constructing, and testing a fully functional CNC robot that saves you 80 percent of the price of an off-the-shelf bot and can be customized to suit your purposes exactly, because you designed it. **$34.95**

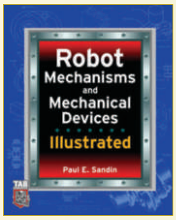
PIC Robotics: A Beginner's Guide to Robotics Projects Using the PIC Micro
by John Iovine

Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU! In this heavily-illustrated resource, the author provides plans and complete parts lists for 11 easy-to-build robots — each with a PICMicro brain. The expertly written coverage of the PIC Basic Computer makes programming a snap — and lots of fun. **$19.95**

Robots, Androids, and Animatrons, Second Edition
by John Iovine

There's never been a better time to explore the world of the nearly human. You get everything you need to create 12 exciting robotic projects using off-the-shelf products and workshop-built devices, including a complete parts list. Also ideal for anyone interested in electronic and motion control, this cult classic gives you the building blocks you need to go practically anywhere in robotics. **$19.95**

Robot Mechanisms and Mechanical Devices Illustrated
by Paul Sandin



Both hobbyists and professionals will treasure this unique and distinctive sourcebook — the most thorough — and thoroughly explained — compendium of robot mechanisms and devices ever assembled. Written and illustrated specifically for people fascinated with mobile robots, *Robot Mechanisms and Mechanical Devices Illustrated* offers a one-stop source of everything needed for the mechanical design of state-of-the-art mobile 'bots. **$39.95**

Robot Builder's Bonanza
by Gordon McComb

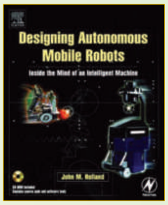
Robot Builder's Bonanza is a major revision of the bestselling bible of amateur robot building — packed with the latest in servo motor technology, microcontrolled robots, remote control, LEGO Mindstorms Kits, and other commercial kits. It gives electronics hobbyists fully illustrated plans for 11 complete robots, as well as all-new coverage of Robotix-based robots, LEGO Technic-based robots, Functionoids with LEGO Mindstorms, and location and motorized systems with servo motors. **$24.95**

Robot Builder's Sourcebook
by Gordon McComb

Fascinated by the world of robotics, but don't know how to tap into the incredible amount of information available on the subject? Clueless as to locating specific information on robotics? Want the names, addresses, phone numbers, and websites of companies that can supply the exact part, plan, kit, building material, programming language, operating system, computer system, or publication you've been searching for? Turn to the *Robot Builder's Sourcebook* — a unique clearinghouse of information that will open 2,500+ new doors and spark almost as many new ideas. **$24.95**

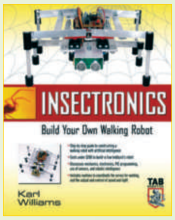
We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

Designing Autonomous Mobile Robots
by John Holland



Designing Autonomous Mobile Robots introduces the reader to the fundamental concepts of this complex field. The author addresses all the pertinent topics of the electronic hardware and software of mobile robot design, with particular emphasis on the more difficult problems of control, navigation, and sensor interfacing. Its state-of-the-art treatment of core concepts in mobile robotics helps and challenges readers to explore new avenues in this exciting field. The accompanying CD-ROM provides software routines for the examples cited, as well as an electronic version of the text. **$49.95**

Insectronics
by Karl Williams



This complete project book delivers all the step-by-step plans you need to construct your own six-legged, insect-like robot that walks and actually responds to its environment. By using inexpensive, off-the-shelf parts, hobbyists can "build a better bug" and have loads of fun honing their knowledge of mechanical construction, programming, microcontroller use, and artificial intelligence. **$19.95**

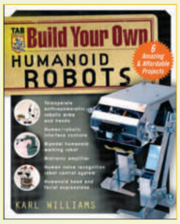
Amphibionics
by Karl Williams

If you're a robotics hobbyist with a flair for creativity, here's your opportunity to join the revolution and advance robotic evolution. This work provides the hobbyist with the detailed mechanical, electronic, and PIC microcontroller knowledge needed to build and program snake, frog, turtle, and alligator robots. It focuses on the construction of each robot in detail and then explores the world of slithering, jumping, swimming, and walking robots — and the artificial intelligence needed with these platforms. Packed with insight and a wealth of informative illustrations, *Amphibionics* focuses on construction details and explores the artificial intelligence needed to make these specialized movements happen. **$19.95**

To order call 1-800-783-4624 or go to our website at www.servomagazine.com

Build Your Own Humanoid Robots

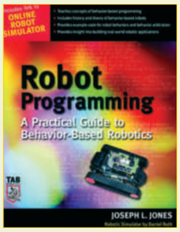
by Karl Williams



Build Your Own Humanoid Robots provides step-by-step directions for six exciting projects — each costing less than $300.00. Together, they form the essential ingredients for making your own humanoid robot. **$24.95**

Robot Programming

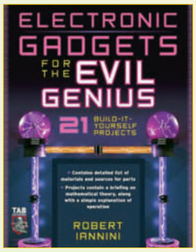
by Joe Jones / Daniel Roth



Using an intuitive method, *Robot Programming* deconstructs robot control into simple and distinct behaviors that are easy to program and debug for inexpensive microcontrollers with little memory. Once you've mastered programming your online bot, you can easily adapt your programs for use in physical robots. **$29.95**

Electronic Gadgets for the Evil Genius

by Robert Iannini



The do-it-yourself hobbyist market — particularly in the area of electronics — is hotter than ever. This book gives the "evil genius" loads of projects to delve into, from an ultrasonic microphone to a body heat detector, all the way to a *Star Wars* Light Saber. This book makes creating these devices fun, inexpensive, and easy. **$24.95**

The Ultimate Palm Robot

by Kevin Mukhar / Dave Johnson

Originally developed by Carnegie-Mellon University robotics department graduate students, this prototype has enjoyed a cult following among enthusiasts. Using software provided by the authors and this step-by-step guide, you can build and operate your own version of the same robot. Learn about parts, software, programming, games, robot resources, and much more from this exciting, one stop guide to Palm robots. **$29.99**

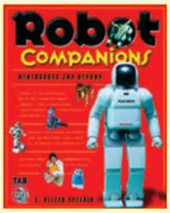
PIC Microcontroller Project Book

by John Iovine

The PIC microcontroller is enormously popular both in the US and abroad. The first edition of this book was a tremendous success because of that. However, in the four years that have passed since the book was first published, the electronics hobbyist market has become more sophisticated. Many users of the PIC are now comfortable paying the $250.00 price for the Professional version of the PIC Basic (the regular version sells for $100.00). This new edition is fully updated and revised to include detailed directions on using both versions of the microcontroller, with no-nonsense recommendations on which one serves better in different situations. **$29.95**

Robot Companions

by E. Oliver Severin



With *Robot Companions*, you'll learn how to build your own robot for purposes such as companionship, supervision of the elderly, tutoring the young, doing household chores, and much more. The book delves into essential enabling technologies — such as mobility, voice, communications, touch, sight, and smell response — so you'll understand the mechanics behind form, function, and personality. **$24.95**

Build Your Own Robot

by Karl Lunt

This book — a compilation of articles from Karl Lunt's long-running column for *Nuts & Volts Magazine* — is a must-read for all beginner- and intermediate-level robotics enthusiasts. It contains entertaining anecdotes, as well as practical advice and instruction. Possible projects range from transforming a TV remote control into a robot controller to building a robot from a drink cooler. You'll want to build them all. **$34.00**

Check out our online bookstore at www.servomagazine.com for a complete listing of all the books that are available.

Robots for Kids

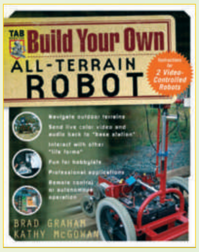
Exploring New Technologies for Learning, First Edition

Edited by Allison Druin / James Hendler

Robots for Kids: Exploring New Technologies for Learning opens with contributions from leading designers and researchers — each one offering a unique perspective into the challenge of developing robots specifically for children. The second part is devoted to the stories of educators who work with children and use these devices, exploring new applications and mapping their impact. Throughout the book, children's essays are provided, discussing their first-hand experiences and ideas about robots. This is an engaging, entertaining, and insightful book for a broad audience — including HCI, AI, and robotics researchers in business and academia, new media and consumer product developers, robotics hobbyists, toy designers, teachers, and education researchers. **$50.95**

Build Your Own All-Terrain Robot

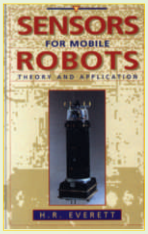
by Brad Graham / Kathy McGowan



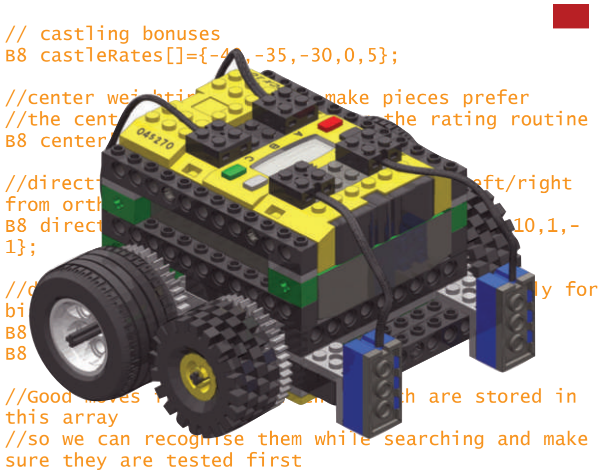
Remotely operated robots are becoming increasingly popular because they allow the operators to explore areas that may not normally be easily accessible. The use of video-controlled technology has sparked a growing public interest not only in hobbyists, but also in the areas of research, space, archeology, deep sea exploration, and even the military. Inside *Build Your Own All-Terrain Robot*, the writers enable even total newcomers to robots to construct a rugged, video-controlled, talking, seeing, interacting explorer bot with a range of over a mile for under $200.00! **$29.95**

Sensors for Mobile Robots

by Hobart R. Everett



In *Sensors for Mobile Robots*, the author compiles everything a student or experienced developmental engineer needs to know about the supporting technologies associated with the rapidly evolving field of robotics. **$69.00**



LESSONS FROM THE LABORATORY

A bi-monthly column just for kids!

— PART 4 —

Line Following Revisited: A Fine Line Between Following and Wandering Off Aimlessly ...

by James Isom

In my last article, we took a look at the fundamentals of line following and programmed a simple single sensor line follower. The cool thing about line following is that it's a simple problem with a variety of solutions. In this installment, we are going to take some time to look at these different methods, using both single and double light sensors. This will also allow us to explore some of the features in our programming environment that we haven't used yet.

If you are still feeling a bit hesitant about programming at Inventor level in Robolab, you can brush up on your skills by viewing my "Robolab Tips & Tricks" page at **www.theroboticslab.com** Props go out to fellow robotics teacher Marc Helfman for helping me compile this list.

As you may remember from our previous program, the robot really didn't follow a line, but rather followed the edge of the line — the point where white met black. Also, our follower could only go in one direction around a circle. If the line were to head off in the opposite direction, our single sensor robot would soon lose track of the line and wander off aimlessly. The goal of this article will be to create a line follower that can follow a line in any direction.

To start, let's take a look at our single sensor line follower program from last issue (Figure 1). What would it look like if we rewrote it in Inventor?

It doesn't look that different, aside from the fact that the steps are strung along one after the other on one screen. The big changes are the "jump" and "land" icons, which replace the pink "run continuously" arrows and the "wait for ..." icons that are specific to waiting for more or less light. They also replace the need for the "greater

Figure 1. Single sensor line follower program.

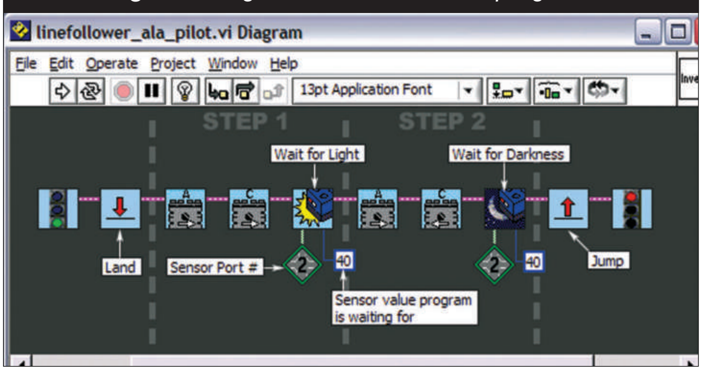
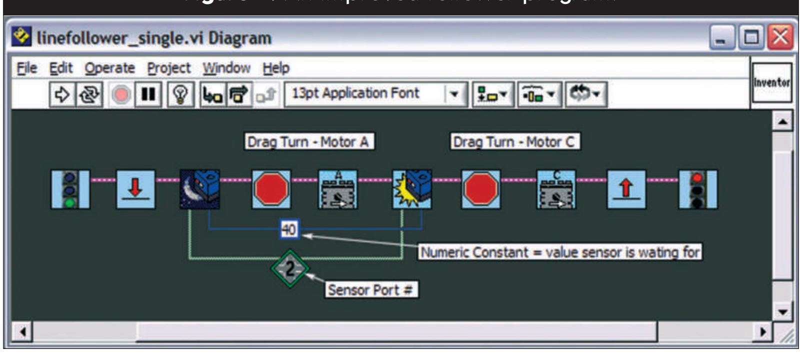


Figure 2. An improved follower program.

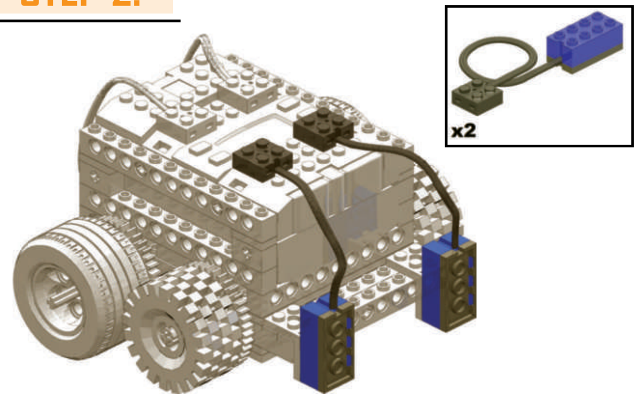


STEP 1:

Parts:

STEP 2:

Parts:



than" and "less than" symbols we used in Pilot.

Of course, this method of line following still has the same problem our program in Pilot did — it can only follow a line in one direction. Let's modify it a bit to see if we can make it follow a line in any direction. Take a look at our second program (Figure 2). With just a slight amount of modification, we can make a line follower that can follow a line in any direction. Our robot still follows the edge of the line, but it does a much better job at traversing one that snakes back and forth.

You may notice that there are a couple of things different about this program. The first and most obvious is that there is only one sensor port identifier (the green diamond with the "2" in it) and sensor value "constant" for both sensor "wait fors." This is really just cosmetic, but — as your programs grow — you'll want to minimize the amount of icons you have on the screen so that it remains easy to read. My mama always said it was good to share and that's true here, too.

The other thing that's different is that we are never going straight, but, instead, we're constantly turning in one direction or the other. Unlike the "point turns" in our Pilot program, which ran the motors in opposite directions, we are performing a series of "drag turns" by only turning one motor on at a time and letting the other side of the robot drag along for the ride. The net effect of this is slow gradual turns that — when run in rapid succession — make for a pretty good single sensor line follower.

Dual Light Sensor Line Following

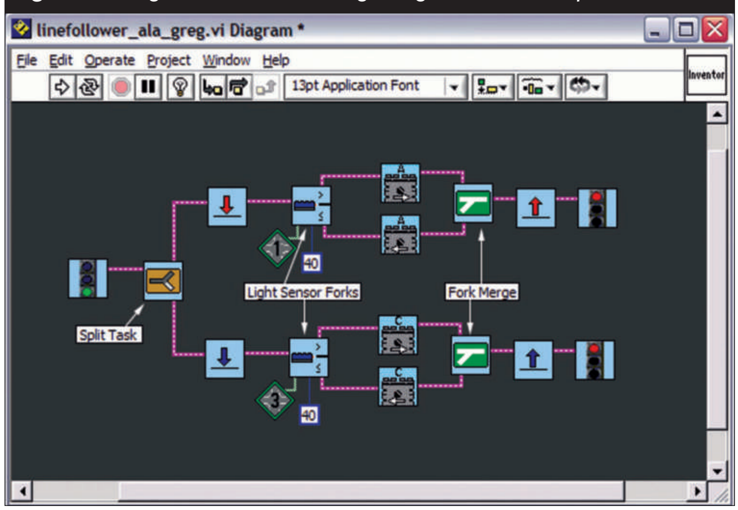
As a veteran of several line following competitions, by far the most common configuration for a line follower is to have two or more light sensors. Take a minute to reconfigure your robot for dual light sensors by following Steps 1 and 2.

The approach to line following with two light sensors might seem a bit obvious; each sensor straddles one side of the line and manages one side of the robot. Indeed, this is the case, but there are still several ways to program the robot to do this. The first method we'll look at was developed by some of my students and involves light sensors connected to ports 1 and 3 (Figure 3).

It looks like a lot is going on here — and there is — but, when we break it down, we'll see that there are really two slightly modified versions of the same program running together, each managing one side of the robot.

Let's start from the green light and move our way left. The program starts immediately with a "split task." A split task allows two things to happen at once, as if two programs were running simultaneously. The upper task inside the red pair of "jumps and lands" manages the left side of the robot or "sensor port 1" and "motor A." The bottom part of the split within the blue "jump and land" manages the right side of the robot, namely "sensor port 3" and "motor C."

Figure 3. Programmed monitoring of light sensors on ports 1 and 3.



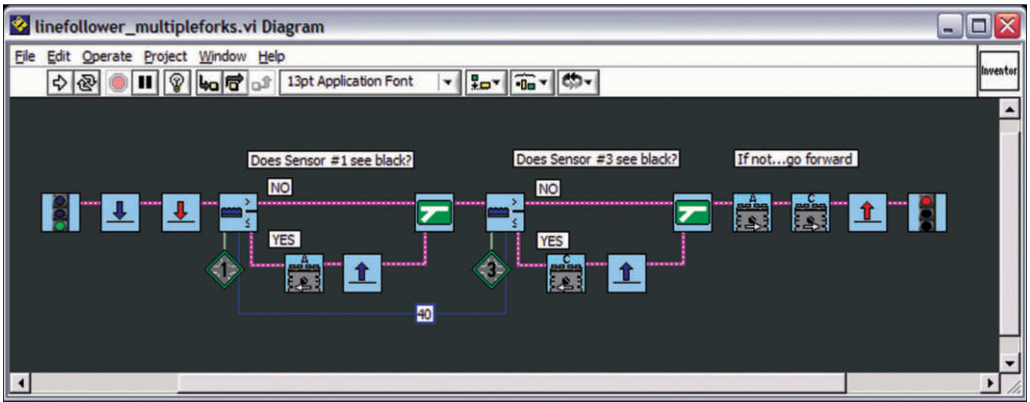


Figure 4. Using sensor forks to detect "black."

Directly after the "land on each task" is a "light sensor fork." A "fork" (there are many kinds) functions exactly like a decision diamond in a flow chart. It does one thing or the other, depending on the value it is waiting for. In our case, we are using light sensor forks, so it is waiting for a value given to it by the sensor.

If the value of the light sensor is greater than 40, it turns the motor on in the forward direction, but, if the opposite is true and the program reads a value less than 40, it turns the motor on in the reverse direction. After the fork makes its decision, it merges again with the program using a "fork merge" and jumps back to the "land" to perform the process all over again. Each fork has to have a "fork merge" before continuing on its way. You will also notice that there are two "stop lights" in our program — one for each task of our "task split." Of course, you may also notice that the program never reaches either stop light because it "jumps" back to its corresponding "land" first.

The combined effect of this two-task program is a pretty robust line follower, with each side being managed by one task in the program. Each "task" doesn't know what the other is doing, but it really doesn't matter in this particular case.

Here are a couple of other ways to achieve the same result:

The one shown in Figure 4 uses a couple of sensor forks to determine whether or not the sensor is seeing black. If either detects black, the program makes a correction by reversing the motor and then jumps back to the beginning. If neither sensor sees black, the robot forges ahead and jumps back to repeat the process.

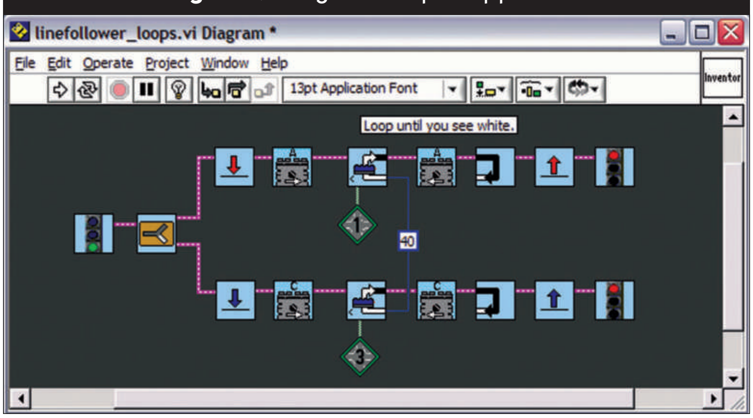
The example in Figure 5 starts with a "task split" and goes forward until one of the sensors sees black. This kicks off a "sensor loop" for that sensor's task that reverses the motor until it's on white again. Loops are often used in programming and are similar to the "jump and land," except that they are smarter — only performing their function until something happens. In our case, the loop only causes the motor to reverse until the light sensor sees white again.

These are just a few of the possibilities for programming a line follower. We now know that there are many ways to approach a single problem. Programming is a bit like painting; lots of people can paint flowers, but each painting will end up looking a bit different in the end.

I suppose you want to know what "the best" way to program a line follower is. That's a bit like asking who makes the better car — Ford or Chevy. There are quite a few variables involved: robot design, nature of the line you're trying to follow, or speed versus accuracy (probably both).

I was recently at a robotics competition where all my students were left in the dust by a LEGO robot sporting three light sensors. So, the jury is still out on this one. If you have a winning line following program using one, two, or three sensors, drop me a line at james@megagiant.com and I'll post your ideas on my website. As always, all of the above programs are available on the *SERVO* website (**www.servomagazine.com**) and my website **www.theroboticslab.com** Happy line following! **SV**

Figure 5. Using a "task split" approach.



Author Bio

James Isom is a part-time robotics teacher and general all-around geek. He has taught robotics to children and teachers in the US and abroad. His website with other additional goodies (including the MLCAD file of this robot) can be found at **www.theroboticslab.com** He can be reached at james@megagiant.com

Where Do We Go From Here?

by Alexa Lindstrom, SERVO Associate Editor

As long as I've been in this field, I've heard about the same vision from every roboticist: One day, in the not-so-distant future, all of us will have personal robots. They will feed our pets, remember the dry cleaning, clean our houses, organize our schedules, and keep us company. For those of us whose first impressions of robots were formed by Rosie the Robot and C-3PO, this dream is very understandable. The problem, however, comes in the implementation. As robot builders, though, the challenge of mapping out a design for a project is second nature; you just have to make sure you're focused on the right issue if you want to take your hobby out of the workshop and make it a daily reality.

Last June, the Business 4Site Summit hosted a round table on personal robots. On the panel were Cynthia Breazeal of MIT, David Calkins of the Robotics Society of America, Fred Nikgohar of RoboDynamics Corporation, and Paolo Pirjanian of Evolution Robotics, Incorporated. Lance Ulanoff of PCMag.com moderated the panel. The topic at hand concerned the form, function, and obstacles involved in the development of personal robots.

So, what is a robotics round table doing at a technology and business summit? Robotics is the hot industry for the future. Perhaps our line followers and sumos only equate to being TRS-80s in the grand scheme of robotic progress, but you have to start somewhere, don't you? The question is where to go from here.

Helen Grenier, president of iRobot, remarked that she cheers every time she hears of the development of a robot that can do back flips or something similar; it reassures her that we (those who are not interested in gymnastic bots) are on the forefront of the field to developing real, practical robots. So, what goals should we be building toward? Is the end result we are striving for closer to the functionoid Roomba, the pragmatic Tetsujin exoskeleton, or the fantastic robots from the movie *I, Robot*? Many would say neither and all, simultaneously.

Paolo Pirjanian's thoughts on this matter clarify the issue: He believes that the rise of robots in society will be subtle and perhaps even largely unnoticed. How can this be? Pirjanian held up a cell phone and stated what would seem to be obvious: A cell phone is a computer. We don't think of it as one; however, if you stop and ask yourself what a computer does, you will find that the cell phone and the microprocessor inside it do all of those things. He believes that the robotic wave of the future will parallel this technological evolution; we will slowly become surrounded by items which are robotic without even realizing they are robots.

If you stop and think, you'll see that we are well on our way to this vision of the future: Our floors and lawns are cared for by robots, our children play with RoboSapiens and MindStorms kits, and even our food is cooled by robots. Got you on that last one, didn't I? As Dave Calkins pointed out, refrigerators qualify under the definition of robots; they sense and respond to stimuli. That is, your fridge can tell when its interior temperature is too low or too high and respond appropriately. Now, Dave went on, would you want your fridge to tell you it's tired today and doesn't want to cool your food?

That brings us to the battleground of robotic development: How human do we want our robots to be? That, I would say, depends on what function you want robots to fill. Roomba is a robot, but many people don't consider it to be one. It vacuums the dust bunnies — who cares what it is? Is Aibo a robot? Most would say yes, but what is the largest function it fulfills? Overall, Aibo and related robots — like the Intelligent Systems pet surrogates — make us feel better. They don't provide a human-like interaction or relationship, yet they provide us with an outlet for entertainment and even caring.

Dr. Breazeal discussed the role of pet surrogates and like products for children undergoing chemotherapy. The robots provide the children with an interaction that places no demands upon them, yet offers a means for the children to feel in control of and responsible for something that interacts on some level with them; this emotional state, in contrast to the rigors of illness and treatment, gives the children a sense of place they otherwise lack. Robots don't need to appear human or act human to connect with and affect us profoundly.

So, should we strive for robots that can interact with us on a higher level? Again, Dr. Breazeal effectively answered this question. She stated that trying to build a robot to simulate a human relationship is misguided. Instead, we should focus on building a robot that is compatible with humans and can complement our lives. To that goal, she and other researchers at MIT are working with computational models, trying to induce robots to share in human emotional states and learn from them.

Such interaction between robots and humans may be more attainable than you would think. We all know MIT's adorable Kismet — the robotic face that responds to human vocal cues to produce appropriate emotional responses. Kismet wasn't even remotely human in appearance, yet people responded to his mechanistic face with gusto.

What made him so endearing? Kismet responded to humans in a way that made us feel that he understood us. He interacted with us; therefore, we feel attached to and intrigued by him. Kismet's successor, Leonardo, goes several steps further. MIT has given him a full body — and some publicity photos even show him fully covered in fur. Leonardo not only interacts with humans, but he learns from those interactions to build new cognitive models.

Leonardo may not speak, but he still communicates effectively to accomplish a task. He can respond to commands with nods or shrugs to signal his understanding or lack thereof. When learning a new task, he will repeatedly look to his human instructor for reassurance. According to Dr. Breazeal, Leonardo learns by gesture, gaze, and action; feedback solidifies any learning about which he is tentative.

In a recorded demonstration that Dr. Breazeal showed at the round table, Leonardo was first taught to turn on and off two colored buttons. After that, more buttons were added and Leonardo reformed his hypothesis, enabling him to apply his learned task to the other colored buttons, too. Throughout the process, he interacted with his human instructor, signaling any confusion or tentativeness he had regarding the task; he also anticipated and responded to praise.

Well, you might say, there's no way I can create something like Leonardo in my shop; I can't contribute to the robotics revolution. Actually, the building blocks for the future of robots are much simpler than Leonardo; he is obviously at the high end of the technology spectrum in the field, but he still can't do one thing: walk.

As Pirjanian mentioned, we all think a robot is intelligent when it can — like Deep Blue — play chess with human champions and win. We want our robots to do all of these cool, smart things. The real test of intelligence, however, is not in the seemingly complex tasks, but rather in the simple ones, like motion. Today's robots face a major challenge in the areas of sensing, movement, and reasoning. This is where the hobbyist can make the strongest impact in the field.

If you get right down to it, for all of the impressive things we can make our robots do, they're still pretty dense when it comes to the things your average three-year-old takes for granted. For example, robots have great difficulty navigating a changing environment; name a robot that could move through a crowded street or even avoid the erratic movements of the family pet. Most robots sense objects around them, but cannot effectively map their own movement within a practical, real world environment.

Motion is a related problem in robotics. Our bots do well, rolling along on the regular surfaces of competition rings or floors, but they cannot navigate stairs or many rough terrains. Particularly since most bots are wheeled, there are restrictions on their movements; legged robots present complex issues of balance and correction. Hexapods, quadrupeds, and balancing robots are beginning to emerge from the hobbyists' workshops; these bots, while non-anthropomorphic, may be the hobbyists' best contribution to the field. Before we can even worry further about the humanoid/non-humanoid issue, we need to enable robots to coexist in and complement human society on a more or less independent level.

Robots, with a few notable exceptions, cannot solve everyday problems. We are still at the stage when low battery sensors and the resultant self-charging are impressive. To effectively interact in a human world, robots must be able to adapt to the alterations inherent in that world. Rosie the Robot may be our dream, but, first, we have to develop the software capability to enable her to distinguish between a stack of important business documents we leave on the table and a stack of junk mail to be disposed of.

Presently, robots have severely limited cognitive ability; Leonardo demonstrates that learning and forming new hypotheses based on interaction is possible. Such programming is currently confined to the research labs, but maybe that is because hobbyists haven't begun to focus on it — yet.

Perhaps we all need to reevaluate our set courses and bring our projects to another level where we aren't worried about following an intermittent line, but rather about avoiding a moving object. Maybe pushing an opponent out of the ring is not as important as programming a bot to distinguish between several others and determine which one is, in fact, the opponent.

The leap from our garage workshops to the cutting edge of the robotic revolution isn't a large one, but it does require that our goals be restructured to answer the real questions, not just the cool ones. **SV**

Advertiser Index

All Electronics Corp. 65	Lynxmotion, Inc. 63	Solutions Cubed 33
Budget Robotics 27	Net Media 2	Sozbots 49
CrustCrawler 23	New Micros, Inc 83	SRS/Robo Magellan 38
Cyberbotics 21	Parallax, Inc. ....Back Cover	Surplus Sales of Nebraska 65
Eagle Tree Systems 40	PCB123/PCBexpress 3	Team Delta 74
E-Clec-Tech 27	Pololu Robotics & Electronics 45	Technological Arts 37
Hack-a-Sapien Contest 71	Robotic Trends 17	Tetsujin 2004 4-5
Hobby Engineering 55	Rogue Robotics 13	Vantec 69

TiniPod™: a very big controller in a very small space.

New $79

The TiniPod™ is a tiny, plug-in board with a powerful brain. The board is only 1.0" x 1.3" and has a very small footprint, plugging into a .2" x 1.2" socket, header, or prototyping area. Connection is made via a 24-pin, dual-row, .1" pins.

The 80MHz, 40MIPs, DSP processor is especially useful in robotics and motion control. The TinyPod™ is capable of driving up to 12 RC Servos at the same time with velocity profiled and acceleration limited moves. The TinyPod™ is also capable of being a 3-axis motion controller, with six channels of PWM outputs and three channels quadrature inputs, all in internal hardware, leaving the processor free to do PID and profiling routines, and forward and inverse kinematics.

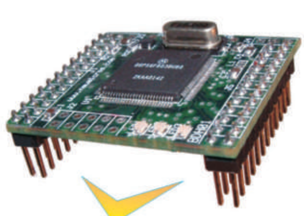
IsoMax™ on the TiniPod™ is an inherently multitasking language with Floating Point math. It is interactive, even allowing debugging while in operation. There's never been anything so small, and so powerful, or so perfect, before.

TiniPod™

- SCI w/RS-232
- SPI
- CANBus
- 6 PWM
- 6 Timers
- 3 LED's
- 64K Flash
- 4K SRAM,

Plug-a-Pod™: plug it in for control!

New $89



Plug-a-Pod™

Here is just the right computer module you want! Plug it in your protoboard, or your own PCB design. You get a 40MIPs DSP-core controller you can program in high level language. The multilayer, high-density, fine-lined, SMT, circuit design is already done for you. Just wire the few extras, drop in the Plug-a-Pod™, program, ... and you're ready!

Make your own 2-layer interface board at a fraction of the cost and complexity of a larger multilayer board with a processor would be. Pick exactly the connectors you need. Add power circuits, isolation, or any particulars of your application. Plug-a-Pod™ fits in less than 2 sq in. The two dual-row .1" pinouts makes connection to your circuit board wiring easy to route. Drop in a Plug-a-Pod™, to bring it all to life.

Feature rich controller: 8-ch 12-bit A/D, 8-GPIO, SCI w/RS-232, SPI, CANBus, 6-PWM, 6 Timers, 3 LED's, 80 MHz, 64K Flash, 4K SRAM, Small C, Assembler, Forth or IsoMax™

One of several award-winning 'Pod products from NMI

If you're serious about robotics and motion control, you must have a 'Pod

Call on the 'Pod's: www.newmicros.com Tel:214-339-2204

Circle #32 on the Reader Service Card.